Reborn BASIC Programming

Platforms
Nintendo DSi / DSi LL
3DS / 3DS LL

PETIT COMPUTER

Petit Computer Official Strategy Technic

Takuya Matsubara
SmileBoom Co, Ltd.
JOLLS INC.

The Heyday of Programming

Introduction to BASIC Again.

There are Practiacl Petit Computer Resources.

Let's enjoy touching the keyboard.

ASCII

Reborn BASIC Programming

PETIT COMPUTER

Petit Computer Official Strategy Technic

Takuya Matsubara
SmileBoom Co, Ltd.
JOLLS INC.

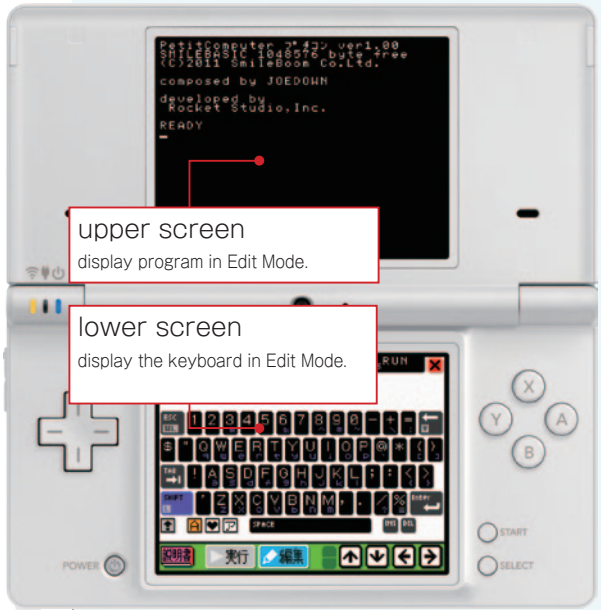
ASCII

PETIT COMPUTER Official Strategy Technic.

CONTENTS

A Brief Guide to the Appeal of Petit Computer

This is PETIT COMPUTER



What is Petit Computer?

Petit Computer brings BASIC right up to date on your Nintendo DSi and 3DS, reviving the classic computer language that introduced many of us to the joys of programming in the 1980s. Once you've got to grips with the basics of BASIC, you'll be able to create images and sounds, and come up with your very own programming masterpieces. Petit Computer is available as DSiWare and runs on Nintendo DSi, DSi XL and 3DS systems. You can purchase DSiWare via the Nintendo DSi Shop or the Nintendo eShop (see p.34 for more information).

Nintendo DSi WARE「Petit Computer」

Platforms	Nintendo DSi／DSi LL／3DS
Developer	SmileBoom Co.Ltd.
Release Date	July 19, 2012
Price	800 Points (see p.35 for more information)

BASIC comes to the Nintendo DSi and 3DS!

Program with the Stylus

You can create your own programs in Edit Mode using the keyboard on the Touch Screen. You can then switch to Run Mode to see your program in action. You can even create your own games by combining pre-loaded music and sound effects with multiple layers of background screens and sprites.

Edit Mode

This mode enables you to enter your program using the keyboard displayed on the lower screen. Your program can be up to about 520,000 characters in length.

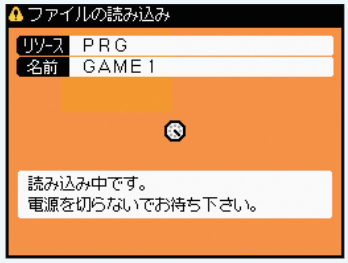
Run Mode

In this mode, you can run programs you have created in Edit Mode, as well as entering commands directly.

Save Your Programs and Exchange Them Wirelessly

For more info, see page 0062

You can save the programs and data you create as files on your system's internal memory. You can then use local wireless communication and enjoy exchanging programs with other users.



▲This image shows a program included with the software being loaded. You can also save your programs.

▲Petit Computer includes a command allowing you to exchange programs with other DSi and 3DS users. (Image: Hiroshi Kuri)

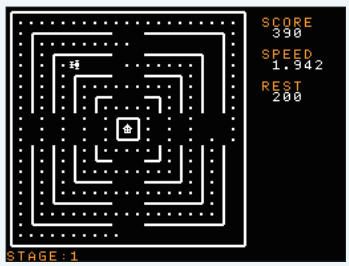
So what exactly is BASIC?

BASIC is a computer programming language first developed for educational purposes in 1964 at Dartmouth College in the USA. Home computers from the late 1970s to the 1980s came with BASIC interpreters as standard, and use of the programming language became near-universal.

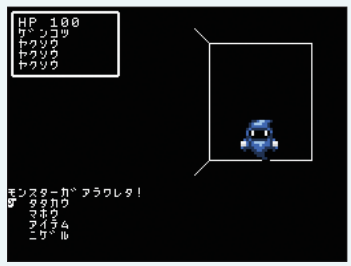
Great Sample Programs are Ready for You to Enjoy

For more info, see page 0042

There are 13 programs included on Petit Computer, ranging from simple introductions to programming to full games, three of which are shown below. They are all programmed entirely in BASIC, so you are free to modify them in any way you choose.



▲Racing Game
FILE NAME「GAME1」
This is a classic old-school game in which you avoid other drivers while collecting dots.



▲Role-Playing Game
FILE NAME「GAME2」
A first-person maze game in which you battle foes and search for treasure chests.

▲Shooting Game
FILE NAME「GAME3」
A game in which you fly through space, blasting enemies. While simple, but it does feature end-of-level bosses.

These cartoon characters are here to introduce you to this all-new 21st century update of BASIC. Some are more serious-minded than others...

Take a look at the official Petit Computer website!

URL : http://smileboom.com/special/petitcom/

You can see other users' programs here!

Check it out! NOW!

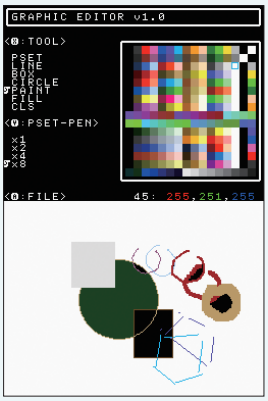
Getting the Most Out of Petit Computer

A Practical Data-Creation Tool

One of the ways Petit Computer differs from the original BASIC is the way in which it allows you to build up multi-layered graphic images from background screens and character images. You'll find that some of the programs included with the software serve as useful tools for creating graphic data. Images from three of these programs are shown here. Make use of them when you want to create your own program using complex graphic images.

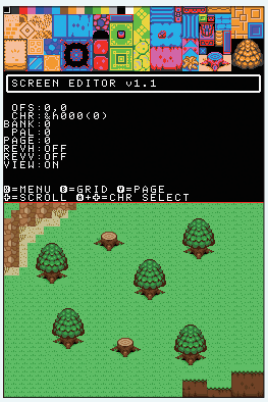
※The BG screen creation tool and character creation tools have been updated on ver.1.1 of Petit Computer, available since June 16th. This guide refers to the updated versions.

For more info, see page 0087



▲Graphic Editor
FILE「GRPED」

▲Character Editor
FILE「CHRED」



▲BG Screen Editor
FILE「SCRED」

People Behind Petit Computer

The President Kobayashi Speaks

This highly-original DSiWare title was developed by SmileBoom, a software company based in Hokkaido in the north of Japan. We spoke to the CEO, Takaki Kobayashi, about how this title came about. Find out just what he had to say to all the Petit Computer programmers out there...

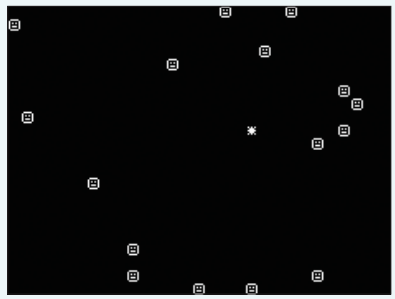
For more info, see page 0018

A Quick Programming Blast

The Single-Screen Programming Challenge

One of the great things about Petit Computer is that you can just pick it up and write your own programs whenever you have a free moment. To show how quick and easy it can be, some of the staff who worked on this guide came up with programs that fit on a single DS screen (24 lines max). The limited space means efficiency is essential. But it's not just technical ability - a single good idea can go a long way. Take a look at some of the results...

▲ A jumping game you can play with a single button. While simple, it makes good use of the sprites included with the software, and even displays your score.



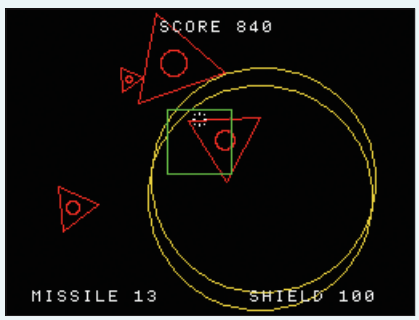
▲ Touch the lower screen to squash the sprites. This demonstrates that even a beginner can program a game that makes use of the Touch Screen.

For more info, see page 0064

In-Depth Programming

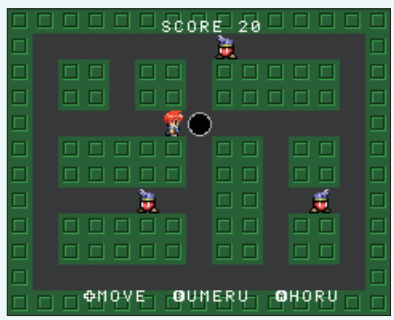
The Epic Program Challenge

In Part 5 of this guide, we will introduce two longer games. Entering the code may be rather time-consuming, but it is worth it in order to learn some more subtle programming techniques. It will hopefully inspire you to flex your programming muscles and come up with epic programs of your own.



SPACE MOLE

Blast incoming foes in this pseudo-3D shooting game. It utilizes the unique capabilities of the DS system with the stylus used to set the target on the Touch Screen.



Heisei Arien

In this old-fashioned action game, you dig holes to trap your enemies before covering them up.

For more info, see page 0106

Welcome to the root of Petit Computer

The Dawn of Home Computing

The 1980s saw the start of a real boom in home computing. Whereas it was once rare to find a device more sophisticated than a pocket calculator in the average home, this all changed as computers became widely-available. Most computers at the time came with BASIC interpreters as standard, and it was perfectly normal for users to make use of programs they had devised themselves. As time went on, computers grew more sophisticated and applications became something you purchased, rather than programming them yourself.

Petit Computer seeks to build on that legacy, giving BASIC a new lease of life, while retaining its original charm.

For more info, see page 0026

Petit Computer's Forerunner

◀"「Family Basic」(1984/Nintendo). A set that taught you how to use BASIC on a home computer. 「Family Basic V3」was sold in 1985."

Turn the Power On, and it's BASIC...

▲Home computers with the MSX BASIC language once reigned supreme. The image is taken from the MSXPlayer emulator and featured in an edition of MSX Magazine published by ASCII in Japan in 2003.

Magazines Contained a Wealth of Knowledge

◀ My Con BASIC Magazine was first published in 1982. It featured programs created by readers. It ceased publication in 2003. The cover image is taken from the October 1984 edition.

▶ An edition of LOGiN magazine, first published in Japan in 1982 by ASCII and by Enterbrain from 2000. It introduced readers to the world of video games until it ceased publication in 2008. The cover image is taken from the August 1985 edition.

The Programmer's Bible

In the early 1980s, 'Konnichiwa Micon' was by far the most widely-read introduction to the world of home computing. It was created by manga artist Mitsuru Sugaya. The hero was a character was a young boy named Arashi, from the hit manga series 'Game Center Arashi'. It was a beginner's guide to programming, and was revolutionary in terms of its accessibility. It inspired a whole generation of programmers in Japan.

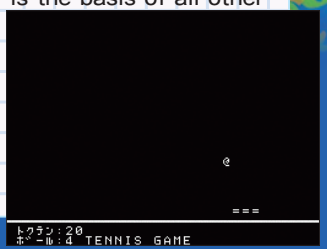
▶ "Konnichiwa Micon'(1982/Mitsuru Sugaya/Shogakukan) Programs featured included an addition program in the first issue, and a tennis game in the second issue."

Stars of BASIC ①

Mitsuru Sugaya

Thirty years ago, when I was busy working on the manga series 'Game Center Arashi', I really got into programming in BASIC on my 8-bit computer. My first original software was a program that calculated tax returns. I wanted to be able to program wherever I was, so I bought a pocket computer and I would be using GOTO and GOSUB commands day and night. I owe my ability to use Java, Perl and Python to that early experience of BASIC. It really is the basis of all other programming languages

Mitsuru Sugaya, who introduced a generation of Japanese programmers to BASIC, has been kind enough to create this original illustration for us. The program in the background can be found in full on the official Petit Computer website. It is actually a port of a tennis game, programmed by a certain Notohoho (http://smileboom.com/special/petitcom/pochette-tennis.html). The image on the right is taken from the game.



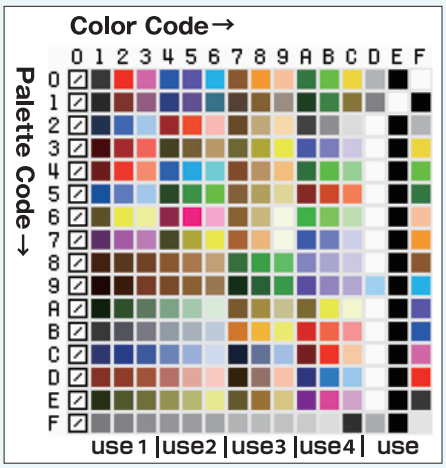
Petit Computer Resources

Petit Computer includes a large amount of pre-loaded graphical data, making it easy to get started and create programs full of visual imagery. This data can be modified using the program's graphic editing tools. In this section, we will introduce some of this graphical data. Refer to it in conjunction with the error code, function and command lists at the back of this guide.

The Console Screen, BG Screens and the Sprite Color Code

The color display on the console screen, the background (BG) screens, and for sprites is made up of units known as palettes, each comprising an array of 16 colors. Each of these colors is assigned a number, from 0-15. This is known as the color code. Please note that the color code 0 is always assigned to a transparent display.

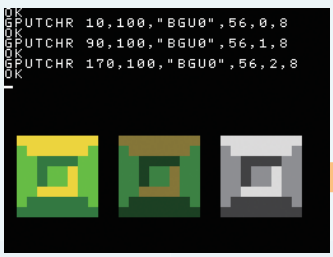
※In this table, the color codes are written across the top, while the palette numbers are written down the side. The numbers are written in hexadecimal (base 16) form.



Graphic Color Codes

The graphic screen can display color codes numbering 0-255 simultaneously. Please see the adjacent table for the order assigned to each color. Colors with the code 0 and 16 are transparent.

※In this table, the color at the top left (00) has the code 0, while the color at the bottom right (FF) has the code 255.



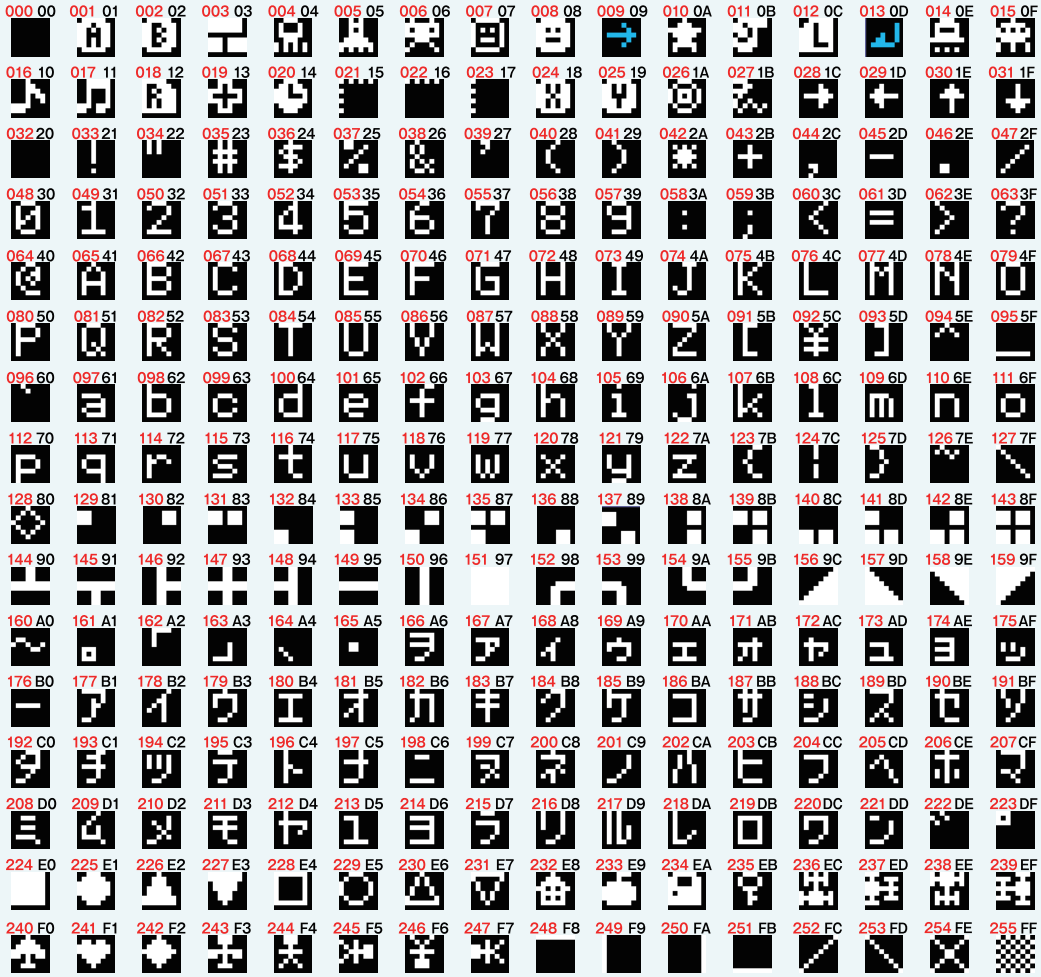
An Example of the Palette Function

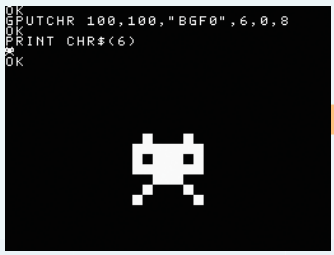
By varying the palette number while displaying the same image, you can completely change the visual impression, as shown in this image.

Character Codes

Character codes are the numerical values assigned in the program to different characters. In Petit Computer, there are a total of 256 different character codes.

※In the table below, the red numbers on the left are decimal, while the black numbers on the right represent hexadecimal (base 16) code.





More info on variables, functions and commands can be found on page 0121

An Example of Character Codes

By using the GPUTCHR and PRINT commands, you can display an assigned character. For instance, the character code 6 will display this character.

● Sprite Characters

A sprite is a character of a fixed size that is displayed on the screen. The name 'sprite' comes from the image of a fairy-like creature that can move freely around the screen. Please refer to the section of this guide that starts on page 79 to learn more about how to use sprites.

For more info, see page 0079

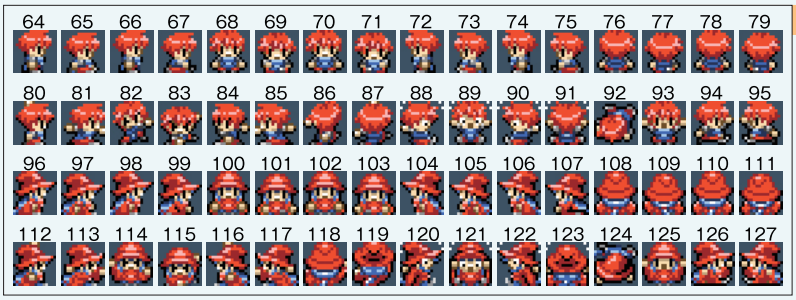
In Petit Computer, there are separate selections of sprites available for use on the upper and lower screens.

Sprite Characters (Upper Screen)

There are a total of 512 different sprites available for display on the upper screen. For ease of use, these are divided into different sets of 64 sprites, each of which is assigned a code SPU0～7. These sprite sets are known as banks.

SPU0

SPU1



SPU2

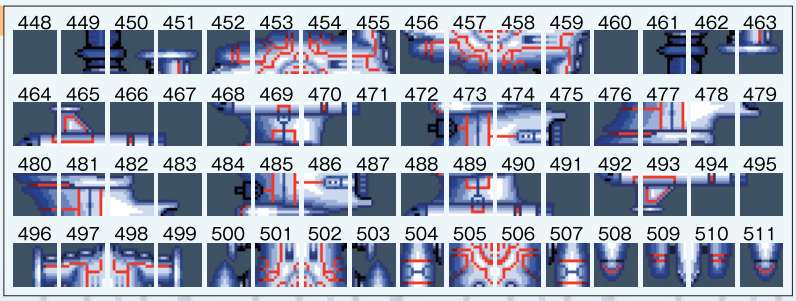
SPU3

SPU4

SPU5

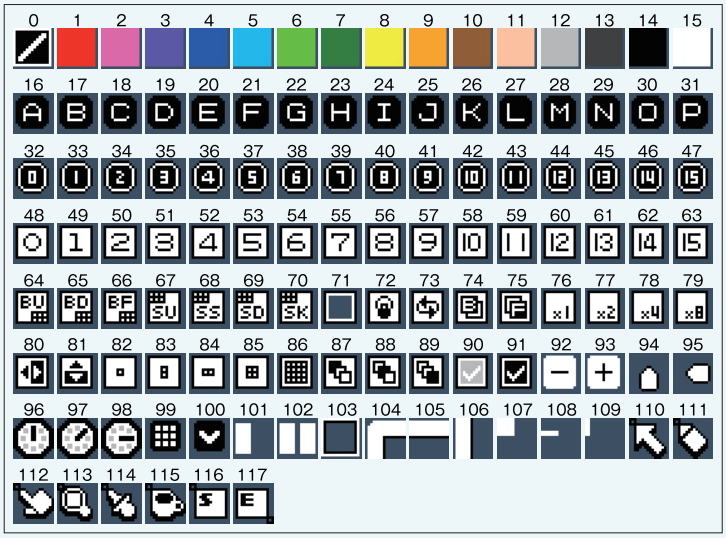
SPU6

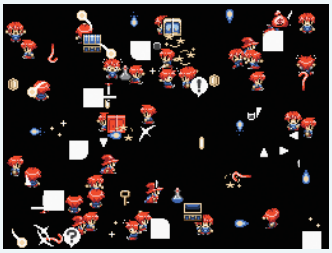
SPU7



Sprite Characters (Lower Screen)

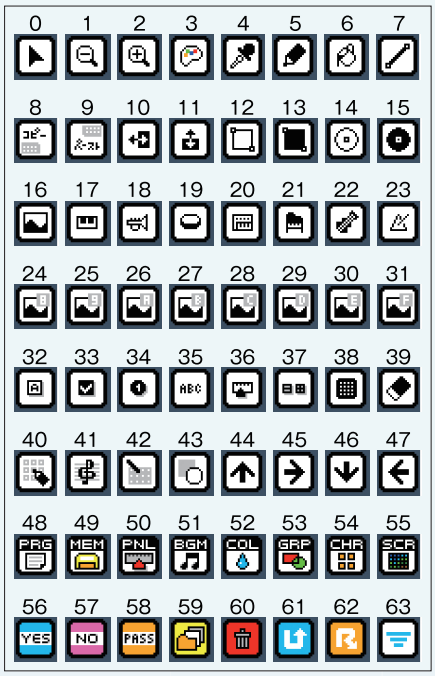
There are a total of 118 different sprites available for display on the upper screen. The code SPS0 is assigned to this bank of sprites.





An Example of the Sprite Display Function

Up to 100 sprites can be displayed simultaneously. As can be seen from this image, each individual sprite can be moved around the screen, in addition to being rotated, expanded or shrunk.

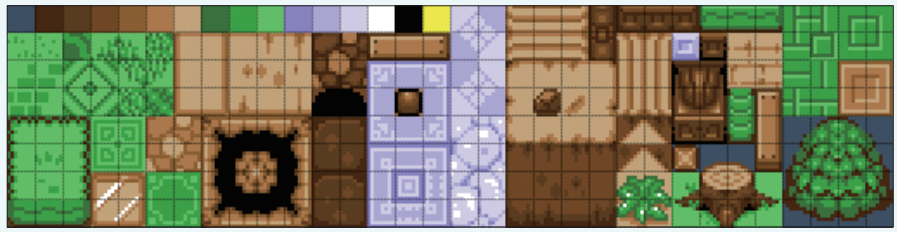


System Icons

There are 64 different user-interface icons which can be displayed on the lower screen. These allow users to select different options, depending on the nature of the program.

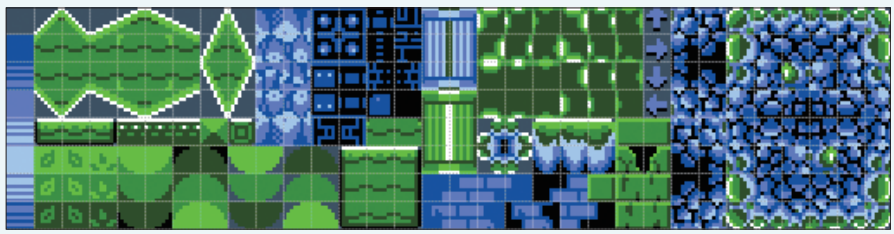
Background Screen Characters

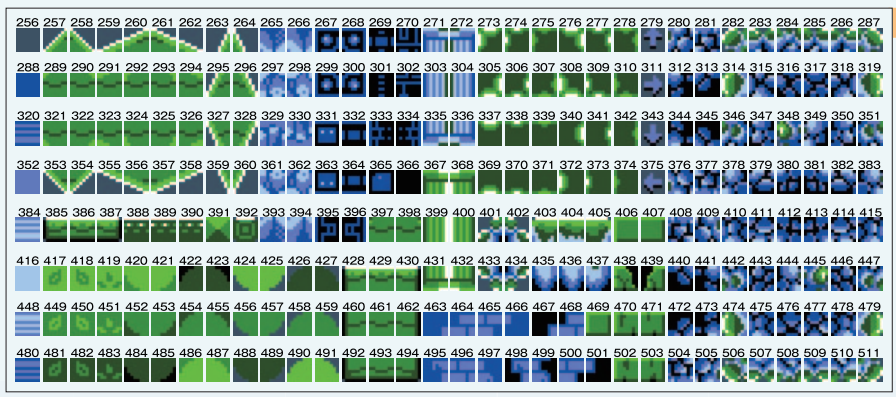
The background screen is abbreviated to the BG screen. It refers to functions relating to background-screen image display. In Petit Computer, there are 4 banks of graphic characters (BGU0-3) assigned to the BG screen.



BGU0

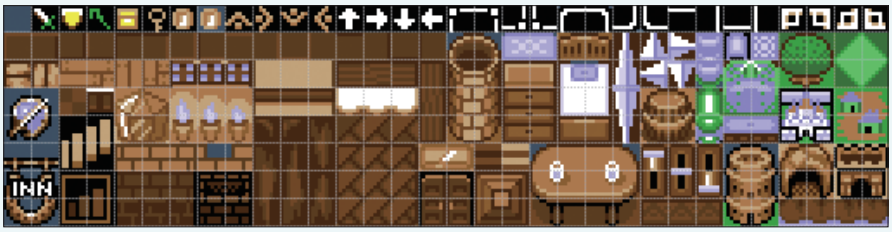
※The example image uses palette 8.





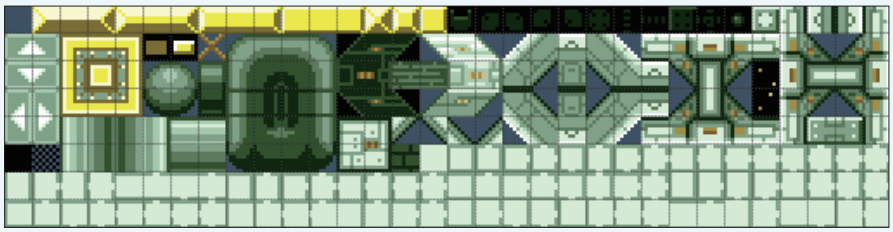
BGU1

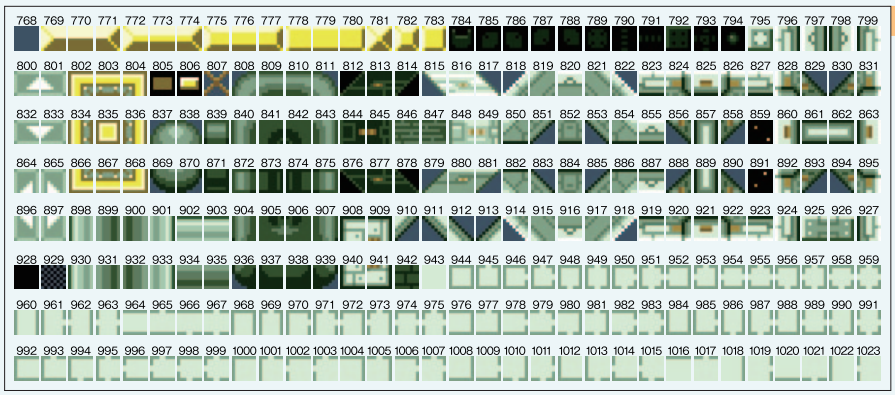
※The example image uses palette 5.



BGU2

※The example image uses palette 8.





BGU3

※The example image uses palette 10.

An Example of the BG Screen Display Function

By creating a background image using both BG screens and positioning a sprite, you can create a classic scene from an RPG. By using images of tree trunks on the rear BG screen, and overlaying another screen with leaves on it, it allows the hero's sprite to conceal itself among the trees.

PART1

So Just What is Petit Computer?

Many of you reading this strategy guide will no doubt have experienced the BASIC boom of the 1980s at first hand. Before we turn to Petit Computer, let's cast our minds back to those heady days and take a whirlwind tour of the history of this classic programming language.

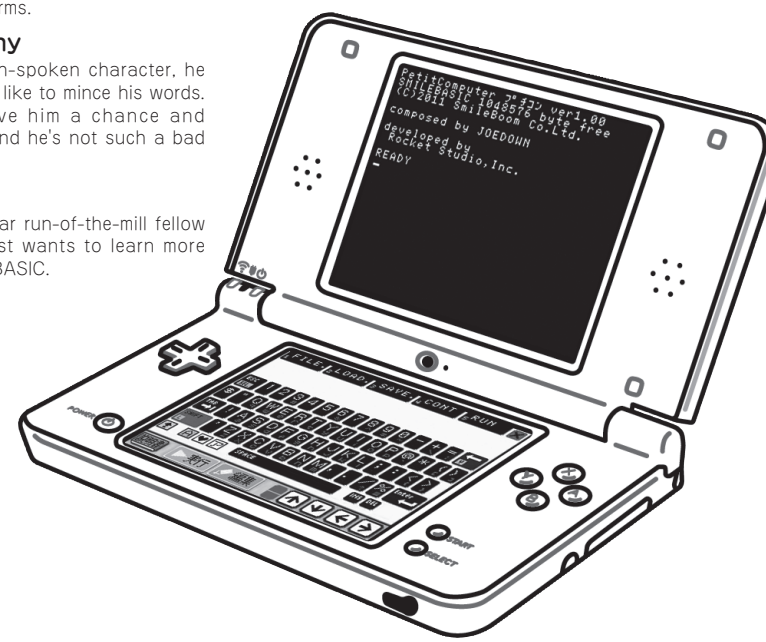
Your Guides to the World of Petit Computer

Professor Jones
Although he looks dry and dull, he was passionate about BASIC in his youth and is prone to reminiscing at length about the good old days.

Nigel
A serious-minded young man, he looks at programming from a technical perspective and loves to explain complex issues in clear terms.

Johnny
A rough-spoken character, he doesn't like to mince his words. But give him a chance and you'll find he's not such a bad guy.

Billy
A regular run-of-the-mill fellow who just wants to learn more about BASIC.



1-01 Meet the People Behind Petit Computer

So how did Petit Computer come about in the first place? What kind of people created it? Was it a labor of love, made by people who experienced the original BASIC boom? We spoke to Takaki Kobayashi, the CEO of SmileBoom, the Sapporo-based development studio who are behind Petit Computer. Here's what we discovered...

From Computer-Crazy Kid to dB Soft...

Before we get started on the main topic of discussion, let's go back a little (okay, a lot...) and talk about how Kobayashi got started in the industry.

Born in the city of Yubari in Hokkaido in the north of Japan, Kobayashi grew up in an area famed for its fossils.

'There was a fossil club at elementary school, and I used to really love collecting them.' But the young Kobayashi's thoughts turned from the past to the future when the home computing boom began during his time at junior high school. He would write programs together with his friends on an NEC PC-8001, a computer with a PCG-series board released in 1970. When he was a senior high school student, he got hold of a Casio computer, the FP-1100, released in 1982. With the hardware being so slow, it gave him the opportunity to learn machine code, and he says that he found it easy to express fun things using computers.

Takaki kobayashi

Born in 1967, he joined DB Software in 1985. Based in Sapporo Hokkaido, he worked on the development of many games. In 1990, he was one of the team who set up Ajenda. In January 2008, he left this company to set up Smileboom, where he is now CEO.

Interview: Takuya Matsubara (Writer and contributor to BASIC Magagine. Worked on Petit Computer for 3 months)
Photographs: Toshiyuki Kurobane

After graduating from high school, Kobayashi joined DB Software, a company based in Sapporo. This was in 1985 when the 8-bit home computing boom was in full swing.

'I actually wanted to go on to university, but having worked on titles like Flappy, I saw that there was a job opportunity in Sapporo. I did an interview, and ended up being hired.'

After joining the company, Kobayashi worked on planning and developing the title 'Woody Poko'. He subsequently worked on games including 'Konyamo Asamade Powerful Mahjang' and 'Melloon'. 'I was really amazed that the company gave me the opportunity to work on titles like this, even though I was a new member of the team.' At the time, DB Software was a company that really liked to create its own development tools and tailor-make DOS programs.

▲'Uddipoko' (DB Software, 1986). An action-adventure game with number of innovative features, including the way time passes, and the use of the left and right hand in the game. Kobayashi worked as designer and programmer for the title. The image is taken from the Windows version of the game, and features in a Japanese publication looking at the classic PC-8801.

▲'Konyamo Asamade Powerful Mahjang' (DB Software, 1988). A huge mahjong game with four different modes. The title display changed with the seasons. Kobayashi worked as designer and programmer for the title. The image comes from the X1 turbo version.

In 1990, Kobayashi set up Ajenda Co. along with five other former colleagues from DB Software. The company president was Mr Fumiya Matsui. For those who are interested, an interview with Mr Matsui and Mr Kobayashi about their time at dB-SOFT is included in a book on PC-8801 classics released in Japan.

All About Studio P

During his time at Ajenda Co., Kobayashi's focus shifted to games for home consoles and mobile gaming. It would be impossible to discuss all the games he worked on here, but one title worth taking a closer look at was 'Studio P', a title released for PlayStation in Japan. This title was a compilation of a number of different tools, with intriguing functions such as one that let you make pots and fire

them in a kiln, or use a special controller called the 'Studio P' to wring out a computer-generated cloth. This was a title where Kobayashi really gave free rein to his individual take on video game technology. 'Well, it's really fun to use new technology in a way that no one would normally think of.' The software included a sequencer and a graphic editor, meaning that it served as a kind of forerunner for Petit Computer.

© AGENDA Co., Ltd.

▶'Studio P' (1996/Ajenda Co.)
Studio P was a dream come true for anyone who wanted to unleash their creativity, featuring everything from 3D animation tools, to sequencers, to paint tools.It included unique ideas, such as letting users make their own pots and fire them in a kiln, or use the special neGcon controller to wring out a wet cloth. It was released for the PlayStation in Japan.

Kobayashi describes how this title brought him to the attention of SCE(Sony Computer Entertainment), who saw that he was creating consistently inventive titles, and they worked together on the cooking-action game 'Ore no Ryori', as well as the board-game themed titles 'Gacharoku' and 'Gacharoku2'.

'Ore no Ryori' was a game that let you experience the fun of cooking using the left and right analog sticks. It was an inventive game with features that included chasing cockroaches out of the kitchen, and catching customers who try to run away without paying. The game is currently available in Japan for PS3, via the PlayStation Network. 'Gacharoku' is a party game in which players can compete simultaneously, rather than taking it in turns. It was released for PlayStation2 in Japan. A sequel in which the action takes place on a larger scale was released the following year.These two titles amply demonstrated that SmileBoom was a company with its own individual philosophy, and that its staff liked to do things others would never think of.

And that's without even mentioning the 'Tkool' series, released by Askii and Enterbrain. Kobayashi was involved in making four 'Tkool' titles between 1998 and 2003, including 'Character Tkool 95'.

Then in 2005, he oversaw the development of 'Daredemo Asobi Taizen' for Nintendo DS. At this point, Kobayashi set his sights on creating a game-creation tool that anyone could use, spurred on by the unflagging desire to create something original and fresh.

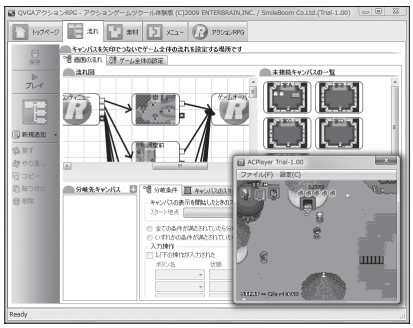
◀ Agenda created a series of game-creation tools, known as the TKool series in English. These included Character Tkool 95 and Shooting TKool 95 which both came out in 1998 in Japan. Other titles are 'RPG Tkool ' , released in 2000, and 'Music Tkook DX', released in 2003. This image is taken from Enterbrain's official Japanese website at http://tkool.jp/
These old TKool titles are taken from the 'Tkook Museum' link on the site.

Making it Easier to Create Games

In 2008, Kobayashi went his own way, and became CEO of his own company, SmileBoom.

While continuing to design and develop games, he also appeared as a speaker at the CEDEC 2008 conference for game developers. Asked whether or not he was going to be making more TKool software, Kobayashi responded by planning and developing 'Action Game Tkool'.

An ambitious piece of software, it followed in the footsteps of an earlier title, 'Shooting Game Tkool', and used newly-developed XNA tools. Its intention was to make games easier to create,and apparently there are games available on Xbox LIVE which were developed using it.



▶ 'Action Game Tkool' (Enterbrain, 2009). The latest Tkool Series title let the user create a series of action games, including shooting games. It was notable for making use of XNA tools, meaning that games created using it were compatible with the Xbox 360. This software could be run on Windows XP and Vista.
The official Japanese site is http://tkool.jp/products/act/top.html

The Ambition Behind Petit Computer

For Kobayashi, the desire to create Petit Computer was born out of his own very personal vision, as opposed to the general direction of the company as a whole. A major influence on his desire to create this game was 'dB-BASIC', developed by DB Software.

'Back then, Hudson had developed Hu-BASIC, and there was also 'Family BASIC'. It felt like any company developing games needed its own version of BASIC. I'd always wanted to come up with my

◀ BASIC developed by DB Software. The image is taken from an integer calculation program, version X1 on dB-IBASIC.

own version, and that dream has finally come true.'

The planning process began with Kobayashi coming up with his own pixel images of the onscreen keyboard, taking them to Nintendo in Kyoto and asking how he should proceed. Various plans were proposed initially, such as using the Wii to program the software, and then transferring it to DSi, or using an FAT file allocation system. Over the course of numerous meetings, adjustments were made to the plans for how to proceed.

▶ The Sharp 'MZ-80B', a machine that Kobayashi admires. He says that if development on it had continued, it may have ended up resembling Petit Computer.

The DSiWare Direction

A crucial issue raised at the earliest stages of planning Petit Computer was that it shouldn't be seen as a means of analyzing or modifying other games and software. From a security point of view, SmileBoom made sure there were no PEEK or POKE commands that could be used to save or load data to the memory, and they prevented user-created programs being downloaded to PCs.

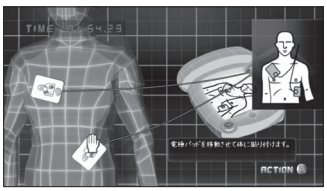
When asked why he decided to develop Petit Computer for the DSi, Kobayashi responded as follows:

'We wanted to make this into a title that people would pick up and try for themselves. As a small company, creating the software as DSiWare meant that we avoided large overheads, and it was really easy to bring it to the market. At the time, the DSi was the only platform that we thought would suit our needs, and in July 2010, we had finalized the design for Petit Computer. It was actually released towards the end of 2010, so it took about three month to complete it.

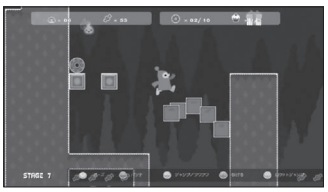
The development for the Petit Computer software itself was overseen by a pair of young developers from a company called 'Rocket Studio', based in Sapporo. Takebe, the president of Rocket Studio, regularly met with Kobayashi, and it was

SmileBoom Welcome to Smileboom

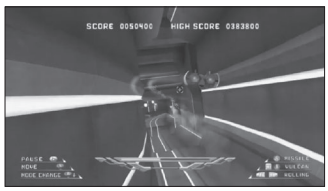
SmileBoom is a company based in Sapporo in Hokkaido, in the north of Japan. It defines its goals as a company as the development of fun games and accessible tools, in addition to consulting, and creating quality online content. Its titles include 'Petit Computer' and 'Action Game Tkool'.



▲ Xbox LIVE Indie Games: 'First Aid'(2009)
This Japanese release teaches people how to use an AED, and perform mouth-to-mouth resuscitation and heart massage. It is popular amongst elderly users.



▲Space Milkman(2009)
This side-scrolling platform game was developed using SmileBoom's 'Action Game Tkool'.



▲Xbox LIVE Indie Games: '3D∞(infinity)'
A shooting game with a 3D display. Even if you don't have a 3D television, it can still be enjoyed using 3D glasses.

Let's meet the other members of the SmileBoom team. The member of staff responsible for single-handedly entering the vast amount of character data required for Petit Computer is called Goto. When Petit Computer was being put together,it was decided that original role-playing, shooting and action games should be included.It was also determined that there should be games with both side-scrolling and top-down perspectives, as well as games with and without gravity. Hosoda was responsible for the optimization of the programs. This made the software's processing speed faster.

The official Petit Computer website was designed by Ueno. His love for computer and video game magazines really comes across in the playful way he designed the site. Kobayashi was in charge of creating the sample programs included on Petit Computer. He actually used algorithms he had developed years previously for the high-speed PAINT command.

◀ Hosoda in Smileboom.He's known as 'the Captain' at work. Was in charge of the fine-tuning of Petit Computer programs.

▶ Ueno in Smileboom.He oversaw design of the official Petit Computer website.

during one of these meetings that the plan to collaborate on this software was hatched. Takabe is a well-known software developer who once worked for Hudson, and worked on the development of 'Family Basic'.

The music for Petit Computer was created by a small team from a company called 'JOEDOWN'. Early in development, the team had to abandon the MML replay function they had planned to implement. But by adding options to control the tone and pitch of the short sounds created using the BEEP command, they managed to create a rich variety of audio content for Petit Computer. Kobayashi's idea was the music should recall the sound of the SCC (sonic custom chip), which would strike a chord with anyone who can remember the classic home computing era. The background music was composed with this in mind, and the team are very proud of what they have managed to achieve.

Rocket Studio, Inc.

▲A software development team from in Sapporo, Hokkaido. They have been involved in a large number of games for handheld devices and mobile phones. Though it is a long-established company with many experienced employees, younger team members oversaw the programming of Petit Computer.

JOEDOWN

▲A music production company from Sapporo, Hokkaido. They have scored a large number of TV programs, commercials and video games. They were responsible for all music and sound effects for Petit Computer.

'=' or '=='

During the development process, there was a great deal of discussion amongst the team about the BASIC functions that should be included on Petit Computer. For instance, a number of functions were revised in September 2010, including the = sign used for IF commands, and the LINE (x, y) format. The team also agreed that IF and ELSE commands extending over multiple lines would make programs hard to read, and so this function was removed.

While it is a little unusual, the ? symbol can be used to stand in for the PRINT command. This was a feature used in the development process that the team intended to remove, but which somehow managed to make it into the final version.

In the planning stages, the intention was that the character creation tool and other functions like this would be fully integrated into the software, and selected using the buttons. However, to reduce

the development workload, someone had the bright idea of actually programming these tools in BASIC. There were also quick commands used, including GPUTCHR which displays character data on the graphic screen, and GFILL which fills a rectangle with one color.

Ideas Make All the Difference

Petit Computer was finally released as DSiWare in Japan in March 2011, for a price of 800 Nintendo DSi Points. According to Kobayashi, this software functions as a promotional tool, showing what the company is capable of, which explains why it is so reasonably priced.

'With the way things are in the industry, the only way to survive is to keep coming up with good ideas. You have to do things other companies aren't doing. You know whether or not you've got a winner at the planning stage.'

The response to Petit Computer was even more positive than the developers had hoped for. When users were asked to contribute their own programs, people would turn up at the Smileboom office, with their DSi in hand. There were also users who sent in programs of incredible length and complexity. SmileBoom even received mails from people telling them that they had bought a DSi XL simply in order to use Petit Computer. The software was initially targeted at users in their 40s, but now Kobayashi wants to give younger people a taste of the joys of programming.

Asked about Smileboom's future ambitions, Kobayashi speaks about his desire to contribute to Hokkaido and make it a better place. He wants his company to explore IT solutions that can be used to improve agriculture and fishing, and strengthen the industrial base of the region. This is a grand vision that goes far beyond the traditional goals of game developers. Petit Computer also looks like it has a bright future ahead of it, and it will be exciting to see how it develops.

To conclude the interview, Kobayashi wanted to share the following words with the readers:

'If those in my generation do not live our lives to the full, how can we expect the younger generation to grow up to fulfill their dreams? This is the time when we need to give it all we've got. Let's try all sorts of things, team up with each other, and do all sorts of crazy stuff!"

(April 2011)

1-02 The Heyday of BASIC

Reading over the interview with the CEO of SmileBoom, it is clear that there is an incredibly strong link between Petit Computer and BASIC culture at the height of the home programming boom. Now, in order to help you get the most out of Petit Computer, let's take a look at the history of home computing and BASIC. It's a huge topic, so we'll just take a tour of the most important points.

The First Home Computer Boom

In 1976, NEC released the TK-80 home computer assembly kit. It really wasn't much more than a stripped-down circuit board, but it was a huge hit. The first Japanese home computer boom had begun. In the wake of this, Japanese consumer electronics companies competed to bring out home computers. Here is a list of some of the main ones...

PC-8801(1979/NEC), BASIC MASTER LEVEL3(1980/HITACHI),
MZ-80B(1981/SHARP), PC-6001(1981/NEC), FM-7(1982/Fujitsu),
X1(1982/SHARP), PC-8801(1981/NEC), PC-9801(1982/NEC)

...etc. etc. By 1984, there were over 50 types of computer on the market, all vying for supremacy.

With this era of computers, it was assumed that users would be able to program, and BASIC came as standard on almost all models. BASIC was a programming language that had originally been created in 1964 for educational purposes at Dartmouth College in the United States. It was developed into what is known as an interpreter (a function that translates and runs commands), and was adopted worldwide. The features of BASIC varied depending on the model of computer, and porting programs between models could be extremely time-consuming.

As an attempt to create a universal programming language, Microsoft and the ASCII Corporation collaborated and released MSX in 1983. Computer manufacturer all released their own models that used MSX, and it came to occupy a dominant place in the Japanese market.

◀ PC-6001mkII'(1983/NEC)
A computer that appeared right in the middle of the boom. It achieved fame for its 16-color display, speech synthesis capability, and high price tag. Its CPU was compatible with the Z80, and it had a 64 kilobyte RAM and VRAM capacity.

Computer Magazines - For Those Who Wanted to Be in the Know

1982 was the peak of the computer magazine boom in Japan. This period saw a shift from the four main magazines ('Monthly Micon'(Denpa Shinbunsha),'I/O'(Kogakusha),'Monthly Ascii'(Ascii),'RAM'(Kosaido)) to new magazines aimed at a younger readership." These magazines included 'Micon BASIC Magazine', fondly known as 'Be Maga', the more entertainment-oriented 'Login', and 'PIO' which was spun out of 'I/O'.

At the heart of these magazines were the type-in computer programs. Computer enthusiasts, keen on saving money by making full use of these programs, put their heads down and entered this code in full. These programs were usually written in hexadecimal machine code, as the priority was to keep the processing speed quick. Naturally, the main way in which data was stored at the time was on cassette. Every old-school computer enthusiast can remember the seemingly-endless screeching sound of those tapes as the data loaded.

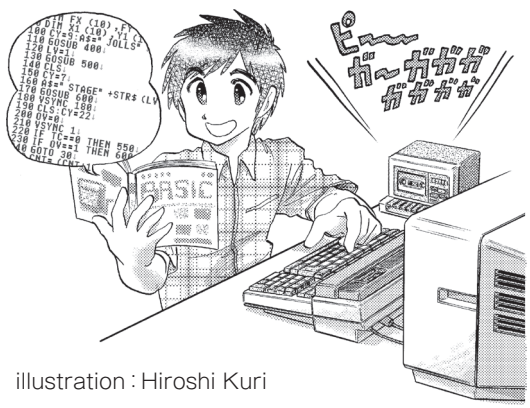


illustration : Hiroshi Kuri

© dempa

Stars of BASIC ②
Hiroshi Kuri

▶'PC Lecture' was published for an impressive 17 years. (Micon BASIC Magazine From Feb 1986)

The earliest incarnation of BASIC had a very restricted memory available for programming, with an 18-digit, 1680 step limit. Yet still, I was able to create several games of my own.

When making a number of games on the PC for 'BASIC Magazine', I tried to replicate the sound that plays at the beginning of a stage in 'Dragon Buster' using the BEEP command, but sadly it didn't sound anything like it... Perhaps I was subconsciously trying to avoid being sued for copyright infringement... With Petit Computer, there are a lot more sound effects, so I get the feeling that you could use this version of BASIC to make a pretty fun music game.

Hiroshi Kuri is a manga artist and illustrator whose primary theme is computer technology. He created the long-running 'PC Lecture' series, which first appeared in 'Micon BASIC Magazine' in 1985. He is an extremely well-known figure among the BASIC generation in Japan.

Home Consoles and BASIC

In some ways, the forerunner of Petit Computer was Family Basic, which was released by Nintendo in 1984 (the third version of the software was released the following year). Family BASIC came with a cassette machine and keyboard which connected to the Famicom, allowing BASIC to be run on it. The main drawback was the restricted RAM.

Many other companies had experimented with BASIC on cassette format, including 'M5'(Sword/Treasure), but it was with the arrival of 'Family Basic' when programming on home consoles really came of age. There are plenty of subsequent examples of BASIC being brought out for home consoles.

For instance, Epoch released software designed as an introduction to BASIC for its Super Cassette Vision console in 1986. In 1996, the game development tools the Debero Box and Debero Starter Kit were released for the PC Engine in Japan by Tokuma Shoten. In 1998, ASCII released Game BASIC for Sega Saturn, and Art Dink released BASIC Studio Powerful Game Workshop in 2001 for the PlayStation 2. Bringing BASIC to home consoles is a project that has always appealed to developers.

The History of N88-BASIC

The most well-known version of BASIC in Japan is probably 'N88-BASIC'. N88-BASIC' began on the PC-8801, before becoming widely-used on the PC-9801 in the N88-BASIC (86) version. There were many different versions, including DISKBASIC, the ROM version of which was launched when the power was turned on, and the MS-DOS version. Some degree of compatibility was maintained between them, which was greatly appreciated by users. N88-BASIC (86) had real staying power, and its ROM version was included on the PC-9821 series released in 1996. This means that although Windows 95 may have been running on the surface, beneath this was a dormant BASIC.

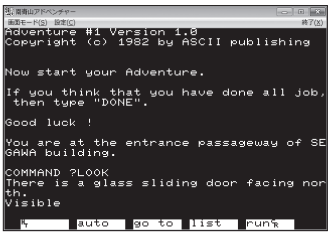
N88-BASIC finally reached the end of its life with the release of the PC98-NX series in 1997, which was PC/AT compatible. While Windows has become the norm nowadays, the DNA of BASIC is retained in software such as Microsoft Visual BASIC and N88 BASIC freeware.

New Game Genres

At the start of the 1980s, the market for computer software expanded enormously, in response to global demand. Programming contests were held offering huge cash prizes, and the popularity of computer magazines spurred on professional and amateur programmers alike to achieve great things.

Computer users were interested above all else in games, which at this time were limited to action and puzzle games. But as home computing developed, so too did the range of gaming genres. This is the period when adventure games, simulations and role-playing games arrived on the scene.

By the mid-80s, floppy disks were widely used to store data, allowing games on a much grander scale, as well as word processing and calculation software. At this time, users' attitudes towards software shifted, and writing your own programs became a minority pursuit. From this point on, BASIC steadily declined in popularity.



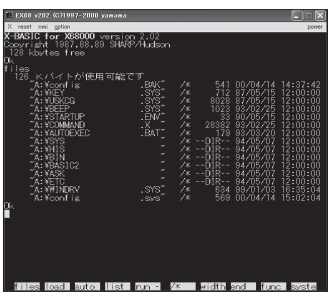
▲'Omotesando Adventure'(1982/Ascii)
This has a strong claim to be first Japanese text-based adventure game. The in-game language was English. The image is taken from a Windows version running on a 8001 emulator, which featured in a publication about the classic PC-9801.

▲The Black Onix'(1984/BPS)
This celebrated RPG put you in the depths of a dark and dangerous dungeon. Making your own map was essential to in the final stages. This image is taken from a Windows version running on a 8001 emulator, which featured in a book about the classic PC-8801.

▲ This classic publication successfully captured the appeal of video games. It was written by Akira Yamashita and published in 1985 by Dempa Shinbunsha. It has been reprinted, along with its sequel about role-playing and adventure games.

The High-Point for Computer Enthusiasts

For computer hobbyists in the 1980s, the top priority was graphical capability. The Fujitsu FM-77AV, released in 1985, offered 4096 color display, but Sharp struck back in 1987 with the X68000, which boasted 65,536 colors. The X68000 came packaged with the arcade hit Gradius, and at the time represented the ultimate in home computing. Fujitsu responded with the FM-TOWNS in 1989, which featured a built-in CD-ROM drive. At the same time, video games designed for business machines such as the PC-9801 began to appear, and became a fixture of the computing market.



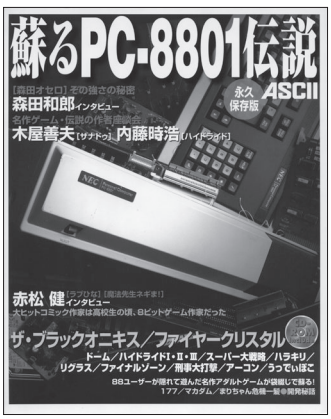
▲ X-BASIC which came packaged with the X68000. A convertor which was sold separately converted code into the C programming language, putting it at the cutting edge of BASIC technology. The image is taken from the EX68 X68000 emulator.

By the mid 1990s, a new generation of home consoles had been released. PCs could not compete with these consoles in terms of price and graphical capability. Consoles seized the initiative with the flagship RPGs that had once been the preserve of home computers, and the PC gaming market began to decline. Petit Computer aims to revive the spirit of that golden period when hobbyists would really make computers their own.

Legends Reborn!

ASCII published a series called 'Yomigaeru Densetsu' ('Legends Reborn'). These captured the spirit and passion of the golden age of home computing. Packed full of features, including interviews with the major figures of the day, as well as CD-ROMs of classic titles, these publications were embraced by fans of vintage home computing. In addition to the editions featured below, we heartily recommend the 2005 edition featuring the NEC 8-bit PC-8001 and PC-6001.

The MSX Magazine collectors' edition from 2002 is also a classic. The second edition was released in 2003, while the third edition came out in 2005.



◀ Legends Reborn - PC-8801 Collectors' Edition was published in 2006. It came with a CD-ROM that contained 13 famous titles, including 'The Black Onix','Hide Lide' 'Woody Poko'.

▶ "Legends Reborn - PC-9801 Collectors' Edition was published in 2004. Including 'Mugen No Shinzo 3' 'Sylpheed' 'Omotesando Adventure' The second edition was released in 2007."

Stars of BASIC ③

Kiyokazu Arai

▶ "From 'Login'(Ascii) 1985 Jan. 'Mashingoshin Kondo' in second square."

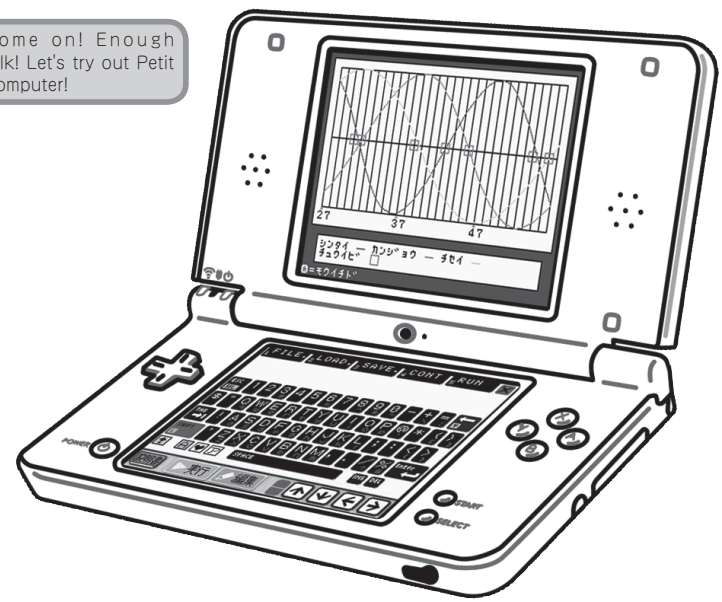
I was responsible for the layout of the computer magazine LOGiN in the mid 1980s. Originally, I drew manga, and when someone asked me to draw a comic strip, I came up with 'BASIC-kun'. He was of course a BASIC-themed character, and his rival was Princess Mashinko Kondo. Her clothes were covered in machine code. I think that nowadays, clothes with programming languages like BASIC written on them might actually be quite a hit. I hope someone follows up on that and actually makes them.

Kiyokazu Arai is the artist responsible for BASIC-kun, a popular cartoon familiar to everyone who was part of the BASIC generation in Japan. It first appeared in LOGiN (ASCII) in July 1984. It subsequently appeared in Famitsu (ASCII/Enterbrain), and MSX Magazine (ASCII). 'Grown-Up BASIC-kun' now appears in Otona Famitsu magazine ('Famitsu for Grown-Ups').

PART2

Getting to Grips with Petit Computer

In this chapter, we give an introduction to Petit Computer, offering a basic overview of its functions for people who have not yet used the software. By studying the sample programs, beginners will be able to get an idea of the possibilities that Petit Computer will put at their fingertips.



2-01 What is Petit Computer?

So what exactly is Petit Computer? Anyone who can remember the days when BASIC was all the rage will probably already have a good idea. But before we give the real thing a go, here is an introduction to this software. To put it simply, Petit Computer is a unique combination of new and old which puts the classic home computers of the BASIC programming era in the palm of your hands.

The Revival of BASIC-Era Home Computing

Petit Computer is a DSiWare program that revives the classic BASIC language that was once synonymous with computer programming. For those of us who remember those days from their youth, computers were very expensive and out of reach of most people. Now, three decades later, BASIC has been revived on the Nintendo DSi and 3DS systems, making this programming language accessible to everyone, young and old.

As the name suggests, Petit Computer gives you a compact home computer in the palm of your hands.

The dual screens of your Nintendo DSi or 3DS system will become the screens of a classic home computer from the BASIC era. The upper screen displays the program list, and functions as the display when your program is run, all in the 8-bit characters familiar from that time. The lower screen features the keyboard display, allowing you to use the intuitive Touch Screen interface to input all kinds of commands. It is just like using a miniature version of the PC-8001 or MSX. Although you cannot save the programs you create to an external drive, you can exchange programs via local wireless communication with other users who have Petit Computer installed on their Nintendo DSi or 3DS systems.

Petit Computer uses its own unique version of the BASIC programming language. It does not differ greatly from the original BASIC language, but features a number of new commands tailored to the functionality of the Nintendo DSi and 3DS systems. These new commands cover a large number of operations, including sprite and background display manipulation, audio and panel components used for input.

Petit Computer also includes 13 sample programs, a huge amount of sample graphic data allowing the display of thousands of different characters, and over 100 varieties of audio data. By making use of this wealth of resources, you can experience the true thrill of computer programming.

Three Screen Sizes

Petit Computer is designed for use with the Nintendo DSi, DSi XL, and 3DS systems. The screen size differs on each of these systems, allowing you to choose the system that best suits you.

Transfer from Nintendo DSi to Nintendo 3DS

By using the Data Transfer tool for the Nintendo DSi, which you can download free from the Nintendo DSi Shop, you can transfer Petit Computer from your DSi or DSi XL to your 3DS.Please note that you cannot transfer Petit Computer purchased on a 3DS to a DSi, and that all save data will be lost when transferring the software.

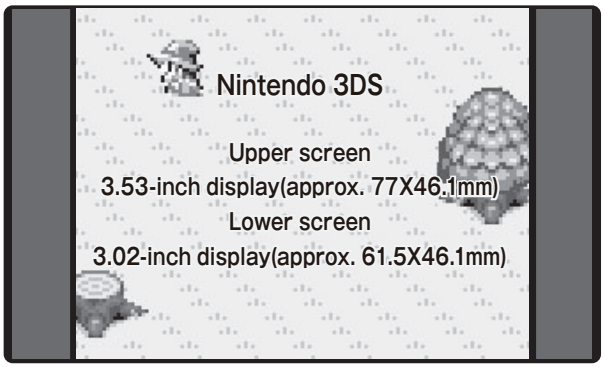
▲(from left to right) Nintendo DSi, DSi XL, and 3DS

Nintendo Dsi XL
4.2-inch display (approx. 85x64mm)

Nintendo DSi
3.25-inch display
(approx. 66x49mm)

Actual Size Display on Different Systems

The image on the left shows the difference in screen sizes between the Nintendo DSi and DSi XL systems. A bigger display is probably better for the input screen for BASIC.For the purposes of using Petit Computer, the DSi XL is recommended.



Actual Size Display on Nintendo 3DS

When running Petit Computer on the Nintendo 3DS, the sides of the upper and lower screens will be black.This is due to the different scaling of the 3DS when compared to the DSi. The resolution is also different, meaning that the enlarged display can make the pixels look a little indistinct. A good tip to switch off this zooming function is to start the 3DS while holding SELECT. This allows you to view Petit Computer in the original resolution.

2-02 Getting Hold of Petit Computer

If you have never purchased DSiWare before, or are not familiar with how the Nintendo DSi works, we hope this guide will prove useful. Needless to say, the Nintendo DSi and 3DS are very user-friendly.However, as Petit Computer is not available in the shops, you may require some assistance when purchasing and downloading it directly to your Nintendo DSi or 3DS.

Here are the simple steps to follow in order to get hold of Petit Computer.

The two absolute essentials are an internet connection and DSi Points.

For those readers familiar with the DSi or 3DS and who knows how to purchase DSiWare, feel free to skip this section. (Please note that all images are of the Nintendo DSi/DSi XL interface, though the interface for the 3DS is almost identical.)

Connecting to the Internet

In order to download Petit Computer to your Nintendo DSi or 3DS, the first thing you'll need to do is connect to the internet. Petit Computer is available for purchase from the Nintendo DSi Shop, or from the Nintendo eShop for 3DS users. Connect to the internet using your home wireless LAN router, or use a special wireless LAN adapter plugged into a computer which is online. If you do not have access to the internet, you may need to contact an internet service provider to find out more about your options.

Once you have enabled a wireless LAN environment, you will need to connect the DSi to the wireless LAN router. To connect your DSi to the internet, first access System Settings on the DSi Menu. Select the Internet option and then select the settings required to connect to the internet access point. See the screenshots below for a clearer idea of the steps in the process.

▲Touch 'System Settings' on the DSi Menu.

▲Scroll over to the third page and touch the 'Internet' option.

▲Now choose 'Connection Settings' to enter the wireless LAN settings. From this point on, the procedure is the same as when you connect any device to a wireless connection.

※ The procedure shown above is for the DSi or DSi XL. When using a 3DS, first access the Nintendo eShop from the menu before following the steps above. Please note that if 'Nintendo eShop' does not appear on your 3DS main menu, you will need to perform a system update. To do this, connect to the internet and select 'System Settings' from the main menu. Select 'Other Settings' followed by 'System Update'.

Another option is to make use of a DS Download Station. These allow users without an internet connection to access the DSi Shop and download Petit Computer.

You can find out more information here.
Check out the official website for a handy guide to DsiWare.
http://www.nintendo.co.jp/ds/dsiware/howto.html

Once you have established an internet connection, follow the steps below to purchase Petit Computer.

Touch 'Nintendo DSi Shop' on the DSi menu. Next, touch 'Start Shopping'. If you cannot access the DSi Shop, check your internet connection and try again.

▲Touch the 'Nintendo DSi Shop' icon on the DSi menu.

▲If it does not connect, select 'System Settings' and then 'Wireless Communications' to check that it is not set to 'Off'.

Obtaining DSi Points

Once you have connected to the internet, you will be able to access the Nintendo DSi Shop and purchase Petit Computer. But before you do that, you need to ensure you have sufficient DSi Points. Nintendo DSi points are a kind of virtual currency required to purchase DSiWare. You can obtain Nintendo DSi Points in the following ways:

- Purchase a Nintendo DSi Points Card at a local retailer or online.
- Access the Nintendo web site via your cell phone (NTT Docomo/au) and purchase a Nintendo Prepaid Code.
- Use a credit card to purchase DSi Points directly from the Nintendo DSi Shop by selecting 'Add Points'.

Normally, the minimum number of points you can purchase is 1000 (if you are using a Nintendo 3DS, you can purchase 500 points from the Nintendo eShop).

Now that you have a supply of DSi Points, you are finally ready to purchase Petit Computer. Touch the 'Nintendo DSi Shop' icon on the DSi menu. Next, touch 'Start Shopping'. If you cannot access the DSi Shop, check your internet connection and try again. Once you have accessed the main DSi Shop page, touch 'Add Points'.

▲Touch 'Add Points' on the main page.

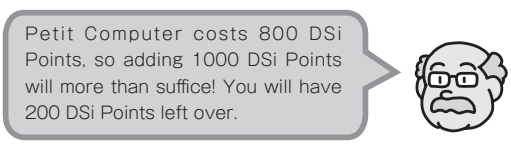
▲You can then choose between redeeming a pre-purchased Nintendo DSi Points Card, or paying with your credit card.

▲Nintendo DSi Point Cards can be purchased from retailers of Nintendo goods. After scratching off the silver area on the back, enter the unique number on the card.

As an example, we have purchased a Nintendo DSi Points Card with 1000 DSi Points. Scratch off the silver area on the back of the card to reveal a number. Select 'Redeem Nintendo Points Card' and enter this number.

Once you have entered the number and received the DSi Points, the card can be discarded. When you add DSi Points, a screen like the one shown below will be displayed. This shows your current DSi Point total. Touch 'OK' to return to the main page.

▲Your DSi Points have been added.

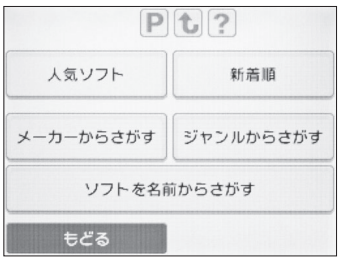


Searching for Software

Now that you've added DSi Points, you'll be able to purchase Petit Computer and other DSiWare. There are a huge number of titles available from the DSi Shop, so you will need to find Petit Computer. That's where the search function comes in. Touch the 'DSiWare' panel at the top of the main page.

Touch 'Find Titles' then 'Search by Software Title'. Use the onscreen keyboard to enter 'Petit Computer'. Finally, touch 'OK' to commence the search for DSiWare with this title.

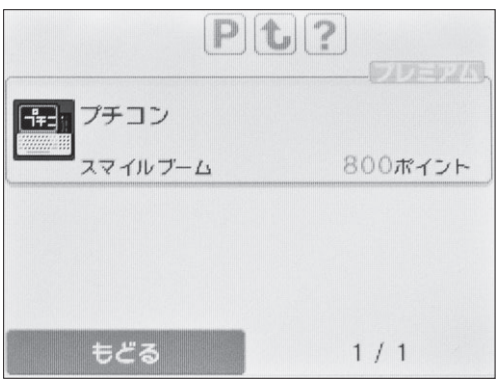
▲Return to the main page and touch 'DSiWare'.



▲Touch 'Search by Software Title'.

▲Touch the onscreen keyboard to enter 'Petit Computer'.

The search results screen should display 'Petit Computer' and 'SmileBoom'. Touch this to proceed.

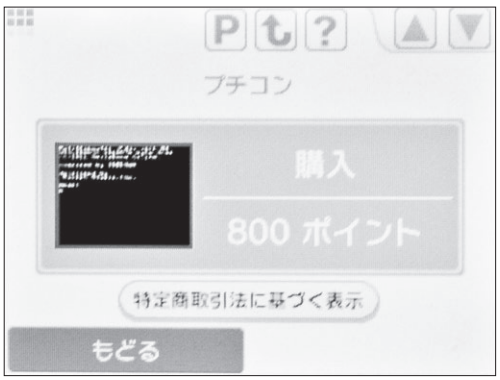


▲Once Petit Computer has been located, touch the icon to proceed.

Downloading DSiWare

Now you're all set to download Petit Computer.

Tap the 'Download' icon on the screen and there will be a message requesting confirmation. Touch 'Yes' to begin the download. 800 DSi Points will be subtracted from your total. Wait while the software is downloaded and saved to the system memory. Petit Computer required 117 blocks. Blocks refer to the amount of memory a title requires. You can install software to a total of 1024 blocks of memory.

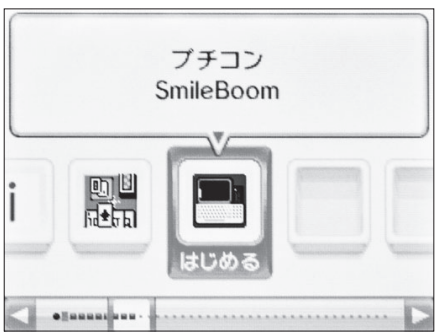


▲Touch the 'Download' panel.

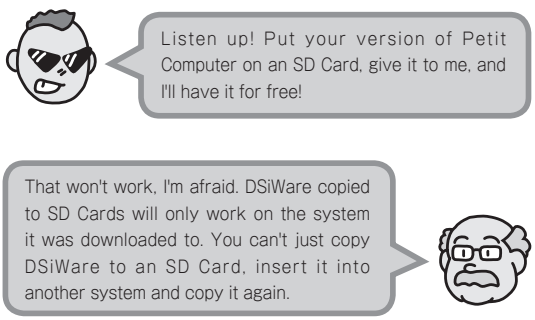
▲When the download begins, the DSi Points and blocks of memory you have remaining will be displayed.

Go back to the Nintendo DSi Menu, and you will find a new icon that looks like a wrapped present. Touch this icon and Petit Computer will appear.

You have now successfully purchased Petit Computer. Touch the icon and let the programming fun commence!



▲Petit Computer has been installed to the DSi. Touch the icon to start.



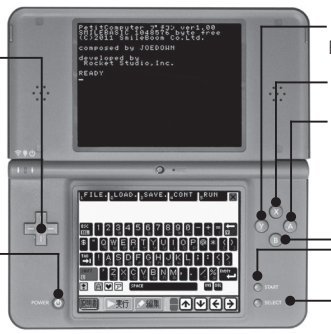
2-03 Basic Controls

Now it's time to experience the fun of Petit Computer hands on! In general, Edit Mode is used to write programs, while switching the screen to Run Mode allows you to put the programs into action. This is different from the way programming works on a home computer, but you'll soon get the hang of it. Now, let's try running some of the sample programs included with Petit Computer and get back to the simple pleasures of BASIC.

The On-Screen Keyboard

Petit Computer combines Touch Screen keyboard controls with input from the DSi and 3DS system buttons. Please see below for a detailed breakdown of the buttons used. The keyboard is displayed on the lower screen, and you can switch between alphanumeric and symbol displays.

Petit Computer Button Control



【+Control Pad】
Moves cursor. Pressing up and down will execute the preceding command in Run Mode.

【Power Button】
Switches the system on and off. It cannot be used as part of Petit Computer programs.

【Y Button】
Backspace

【X Button】

【A Button】

【B Button】

【START】
Run program

【SELECT】
Stop program

【R Button】

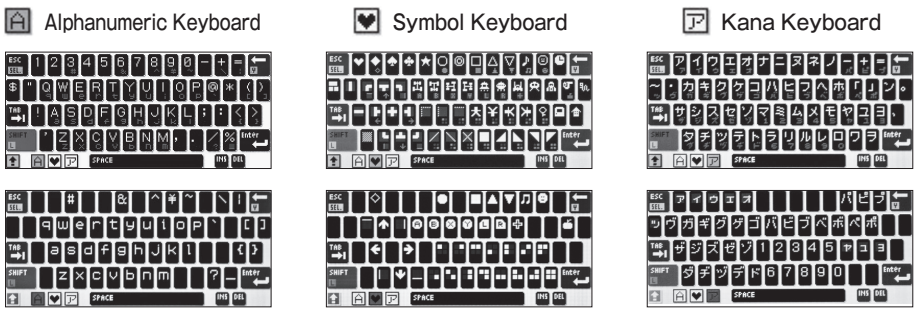
【L Button】
SHIFT key

【L Button + Left/Right on cursor】
Skip page

【L Button + Left/Right on cursor】
Move to top or bottom of page

【L Button + Y Button】
Delete a line of a program

Touch Screen Keyboard



The Three Modes

Petit Computer has three screen modes: Run Mode, Edit Mode and the Manual, which displays usage instructions. By pressing the appropriate button on the lower Touch Screen, it's easy to switch between these modes. When first starting Petit Computer, you will begin in Run Mode.

① **Function Keys**: Allows you to select functions by pressing a single key. In Run Mode, these include useful commands such as FILES, LOAD", SAVE", CONT and RUN.
② **Close Button** : Exits Petit Computer and returns to the main Nintendo DSi or 3DS menu screen. Press this button and you will be asked to confirm whether you wish to exit. Select 'Yes' to proceed.
③ **Help Button** : Opens a screen displaying usage instructions.
④ **Run Button** : Switches to Run Mode and runs the program. While programs are running, it functions as a PAUSE button.
⑤ **Edit Button** : Switches to Edit Mode.

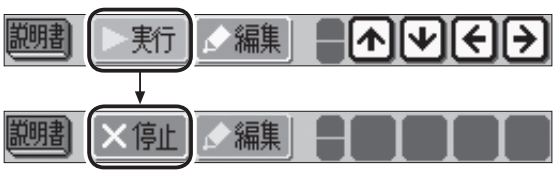
Please note that in addition to ② above, you can also press POWER to exit Petit Computer. Be aware that a confirmation message will not be displayed in this case, and that if you keep POWER held down, the system will shut down. The commands assigned to function keys in ① above can be changed using a KEY command. Refer to the Petit Computer Resources section at the end of this guide for more information.

●Run Mode

This mode allows you to run the programs you have created in Edit Mode. In addition, when no program is being run, you can enter special Run Mode commands of up to 32 characters (see page 52). Remember to always include ENTER at the end of your commands. If the command can be run correctly, an OK message will be displayed. If it cannot be run, an ERROR message will be displayed.

Up/Down on +Control Pad : **Input preceding command**
Left/Right on +Control Pad : **Move cursor left and right**
Y Button : **Delete character to left of cursor**
START : **Run program (same as RUN command)**
Left on +Control Pad + R Button + START : **Reset screen. A useful command when the screen display becomes difficult to read.**
SELECT : **Stop program**

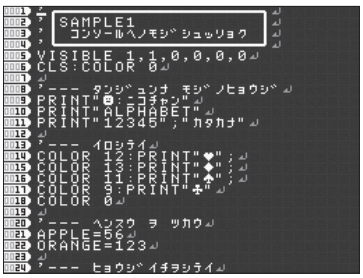
While programs are running, the RUN button will operate as a PAUSE button. Use this to pause programs mid-way through.



●Edit Mode

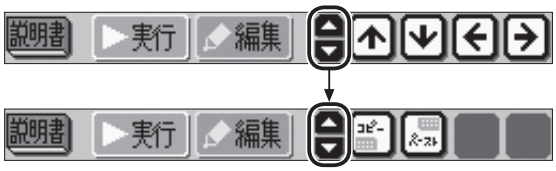
This mode allows you to write and edit programs. Unlike in Run Mode, you can enter programs over multiple lines. The maximum number of lines for a program is 9999, with a limit of around 100 characters per line. (The maximum number of characters in a program is approximately 520,000.)

Note that programs you are currently working on will be lost if you switch the system's power off, so be sure to use the SAVE command to store your progress.



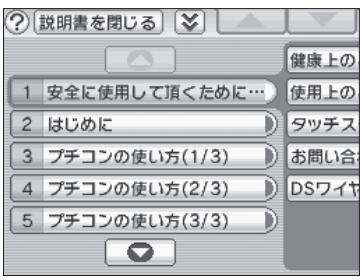
+Control Pad : Move cursor.
Y Button : Delete character to left of cursor.
+Control Pad + L Button : Move cursor to next screen or preceding screen. Useful for navigating longer programs.
START : Switch to Run Mode and run program.
System Icons : These are the cursor icons displayed at the bottom right.

These icons can be used in Edit Mode to move the cursor. You can also press ▼ to use the copy-paste function. Pressing COPY will copy the entire line from the cursor's current location. Pressing PASTE will insert this line directly after the cursor.



●Manual

This allows you to access the usage instructions for Petit Computer, giving explanations of the software's key functions, system variables and commands.



2-04 Check Out the Sample Programs

The real joy of Petit Computer is creating your very own programs. But before you come to write your own programs, it's good to get a feel for the software with the sample programs that are included with Petit Computer. Before giving a brief overview of these programs, let's look at how to run them.

Commands for Reading Programs

The EXEC command tells the system to read a program from the file and run it. So, for instance, if you want to run SAMPLE1, you would input the following command on the keyboard in Run Mode.

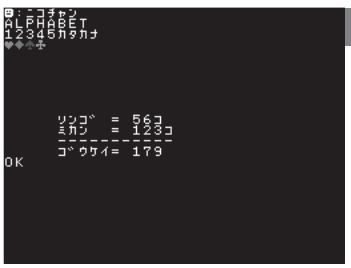
```
EXEC"SAMPLE1"↵
```

By modifying the text in inverted commas, you will be able to run other programs (SAMPLE1-7). You could also use the LOAD and RUN commands to perform the same operation.

If you run a number of programs in succession, the on-screen display may become corrupted. This is because data from a previous program has not been deleted, and is affecting the program that is currently running. If this occurs, you can either restart Petit Computer, or refer to page 49 for more information on how to reset data.

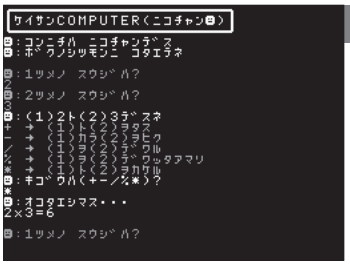
Sample Programs (Elementary Samples)

Petit Computer includes 13 different programs written in BASIC. In this section, we will look at the 7 most elementary programs. The contents of these programs are not overly complex, and beginners to BASIC should be able to grasp how the programs function by taking a look at the program list in Edit Mode.



● FILE 'SAMPLE1'

This is a program that displays an array of characters. It is a very short program, and 'OK' is displayed to indicate that it has finished running. It is a great first step for a would-be programmer as it demonstrates the fundamentals of text display and variable usage.

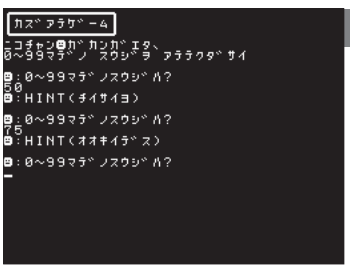


● FILE 'SAMPLE2'

This is a simple calculation program, making use of character input. By inputting a numeral in response to the first and second questions, and then inputting the symbol for a mathematical function (+ - / % *) in response to the third question, the result will be calculated and displayed. For example, if you wish to calculate 2x3, you would touch '2', '3', and '*' on the keyboard. Take a look at the program and you'll see how a result is calculated from the inputted numbers.

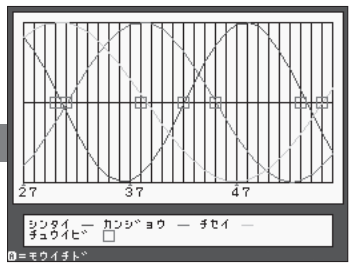
● FILE 'SAMPLE3'

This is a simple synthesizer program controlled via the keyboard. You can also press the +Control Pad to play cymbals. It serves as a good example of how button input data is processed, and how audio samples are played.



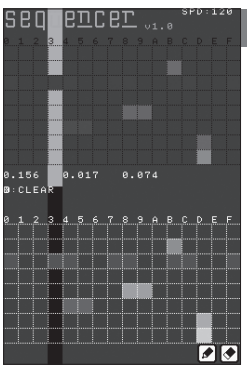
● FILE 'SAMPLE4'

This program is a number guessing game. By entering a number between 0 and 99, you will try to guess the number the computer has generated. You will be given hints as to whether the number you entered is greater or smaller than the correct answer. The program includes commands for generating random numbers, as well as branch instructions for comparing the number you entered to the original number the computer generated.



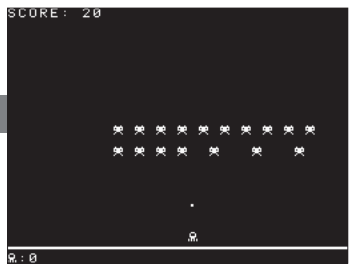
● FILE 'SAMPLE5'

This is a program designed purely for fun which gives false biorhythm information. Enter your birthday and the program will generate graphs showing information on your body, emotions and intelligence. It utilizes the SIN function to calculate and create curves.



● FILE 'SAMPLE6'

This program lets you use a sequencer with up to 8 tracks played simultaneously. By touching the lower screen, you can select the sound you wish to be played. Pressing left and right on the +Control Pad will adjust the tempo. Pressing the X Button will clear the data you have entered.

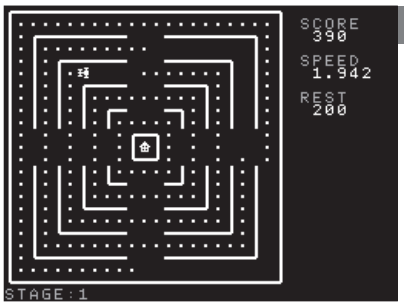


● FILE 'SAMPLE7'

This program replicates a classic alien invader game. Press left and right on the +Control Pad to move your craft, and touch the lower screen to fire at your alien foes. This program provides a useful reference for manipulating multiple sprites, as well as collision detection. Please note that this is not a complete game, as there is no collision detection for when the enemy missiles hit you.

Sample Programs (Games)

In this section, we will introduce 3 of the game programs that come included with Petit Computer. Run these games in the same way described in the last section, by entering EXEC "File Name". For users with programming experience, please feel free to modify the source code and see what you can come up with. Please note that you cannot overwrite the original program with your modified version, but you can save it under a different file name.

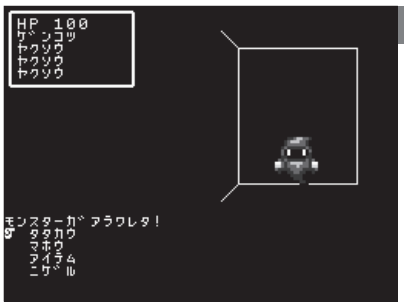


● RACING GAME (FILE"GAME1")

This game is known as 'Dot Racer'. You control a car and your aim is to collect all the dots that line the course while avoiding rival cars. You can use the +Control Pad to change the track you are racing around. Be careful, as the more dots you collect, the faster you will go.

Advice of improvement

Line 222 of the source code relates to the speed of your car. If you reduce the amount that is subtracted during the speed calculation, your car will accelerate less rapidly.



● Role Playing GAME (FILE"GAME2")

This is a first-person dungeon crawling game. You will explore a labyrinth while taking out enemies. If you complete all 3 levels, the game will end.

Advice of improvement

Line 72 of the source code contains the calculation for random encounters with enemies. Line 390 determines the damage inflicted on the player. Line 403 determines the damage inflicted on foes. If you are finding the game difficult, you can make it easier by adjusting the values in these lines.

● Shooting GAME (FILE"GAME3")

This simple shooting game utilizes the sprite function. It even features end-of-level bosses.

Advice of improvement

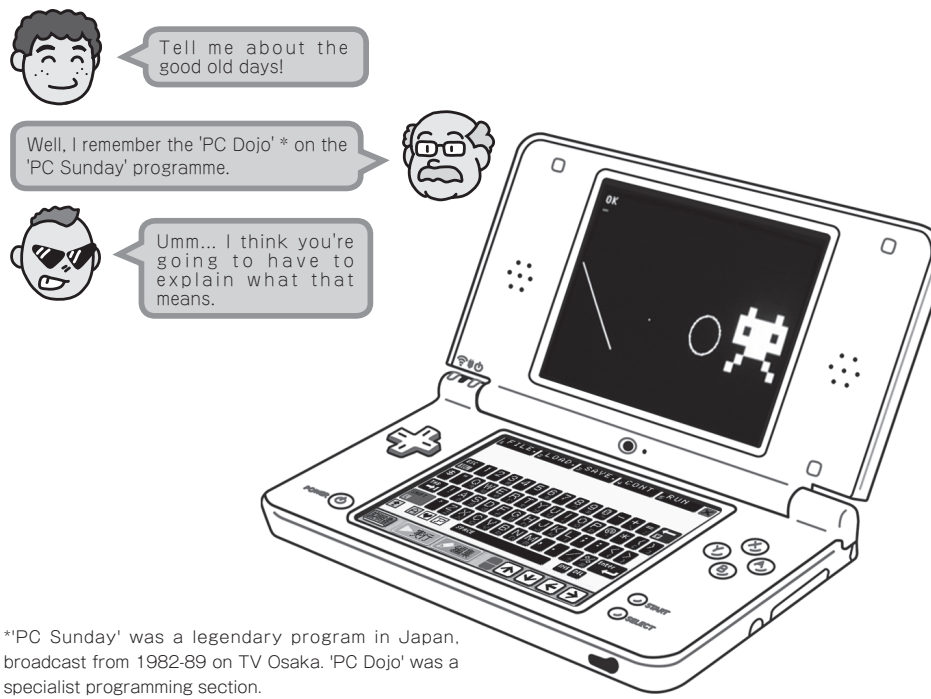
Modifying line 361 of the source code as shown below will make enemy bullets all fly towards you (with the exception of the bosses' bullets).

```
ENSHDIR(ESHTCNT)=ATAN(MYX-ENMX(I)
,MYY-ENMY(I) )
```

PART 3

Learning to Program with Petit Computer

In this chapter, we will look at some simple programming techniques. You will be able to get an idea of the true fun of Petit Computer as you learn to create the images and sounds you want by simply tapping the keyboard.



*'PC Sunday' was a legendary program in Japan, broadcast from 1982-89 on TV Osaka. 'PC Dojo' was a specialist programming section.

3-01 The Basics of BASIC

Petit Computer uses its own unique version of the BASIC programming language. It uses the latest technology to revive the BASIC language that many of us remember so fondly.

The way that programs are written does not differ fundamentally from the original BASIC, but there are a few minor differences. For instance, on Petit Computer, there is no need to enter line numbers. You can simply add labels to indicate which lines GOTO and GOSUB commands should go to. There are also a number of functions and commands specifically for use on the Nintendo DSi and 3DS. Users who are familiar with BASIC should pay special attention to these new elements.

The Commands Everyone Should Know

If you are reading this guide, there is a good chance that you are already familiar with BASIC to some degree. Perhaps you think that you have no need for a refresher course. But it never hurts to go back to first principles, to remind yourself of the commands without which you cannot program. Whether it's your first time programming, or you have not programmed in years, it is well worth reading this section.

Petit Computer contains sample programs for reference, which were introduced in Part 2-03 of this guide. It is a great idea to take a look at these programs' source code as the first step towards writing your own programs.

● Assign & Output

A value or character string can be assigned to a variable with an equals sign (=), and this can be outputted by using the PRINT command. Even if you are already familiar with this, it is always worth going over the absolute basics again.

The most simple sample program included with this software is the one which displays two values and then calculates and displays the final total. This is SAMPLE1.

From sample program 'SAMPLE1'

```
APPLE=56                          ←The number 56 has been entered for the variable 'APPLE'.
ORANGE=123                        ←The number 123 has been entered for the variable 'ORANGE'.
:
LOCATE 5,10                       ←Display coordinates (for on-screen location).
PRINT"ﾘﾝｺﾞ  = ";APPLE;"ｺ"         ←Displays value entered for 'APPLE'
LOCATE 5,11
PRINT"ﾐｶﾝ   = ";ORANGE;"ｺ"        ←Displays value entered for 'ORANGE'
:
LOCATE 5,13
TOTAL=APPLE+ORANGE                ←The two variable values are added together and
                                   assigned to the variable 'TOTAL'.
PRINT"ｺﾞｳｹｲ= ";TOTAL              ←Displays value for variable 'TOTAL'.
```

Lines 21 and 22 of the source code relate to the variables that give the number of APPLES and ORANGES. Lines 25-28 relate to the different values needed for the calculation, while lines 31-33 display the total.

Line 26 uses a PRINT command to display the number of apples. You will note that some terms appear in the program in "speech marks", while the word APPLE does not. The speech marks tell the program to treat the text contained within them as character strings - this means they are displayed on-screen precisely as they appear in the program. In contrast, the word APPLE, which is not contained within speech marks, is treated as a numerical value, and the value assigned to it in line 21 should be displayed (56).

In line 32, both values are added together in order to calculate the total. This total is a new variable that is assigned by using TOTAL. The LOCATE command indicates the coordinates where the characters should be displayed on-screen. To learn more about how to display characters on-screen, please refer to Part 3-02 'Displaying Characters', on page 53.

In earlier versions of the BASIC programming languages, the equals sign '=' was often used both for assigning values, as well as for indicating whether or not a value is equal (as what is known as a relational operator). In Petit Computer, a single equals sign '=' is used to indicate that a value has been assigned to something, while a double equals sign '==' is used to indicate whether two values are equal or not. For more details, please refer to the section on relational operators on page 51. To clarify, if you write 'A=1' in a program it means that the value '1' has been assigned to the variable A. However, if you wish to compare the values 'A' and '1' to see whether or not they are equal, you should write it with a double equals sign: 'A==1'.

●Conditions & Branches

In order to calculate whether or not specific conditions have been met, you will use an IF～THEN operation. If you wish to make the program go to a certain line or section, you will use a GOTO command. A subroutine is a self-contained section within the source code which you can get the program to repeatedly return to. To go to a specific subroutine, you will use the GOSUB command, while the RETURN command will designate the line of the program to go back to. Petit Computer programs do not use line numbers. Instead, labels are assigned to the lines where you wish GOTO and GOSUB commands to go to. Use @ along with the label name to indicate a line.

To see a simple program where IF～THEN and GOTO commands are used, please refer to the sample program, SAMPLE2. In this program, 2 numerical values are entered, along with the symbol for a basic arithmetical operation (+ - / %*) and the final result is displayed.

From sample program 'SAMPLE2'

```
0013 @LOOP
  ⋮
0015 INPUT"😐:1ﾂﾒﾉ ｽｳｼﾞﾊﾝ";NO1      ←Value of inputted button is assigned to variable 'NO1'.
  ⋮
0018 INPUT"😐:2ﾂﾒﾉ ｽｳｼﾞﾊﾝ";NO2      ←Value of inputted button is assigned to variable 'NO2'.
  ⋮
0031 INPUT"😐:ｷｺﾞｳﾊﾝ(+-/%*) ";K$   ←Character of inputted button is assigned to character
  ⋮                                  variable K$.
0034 MARK=0                        ←0 is assigned to variable 'MARK'.
0035 IF K$=="+" THEN MARK=1        ←If the inputted key is + then the value of the variable
  ⋮                                  'MARK' changes to 1.
```

✚ To Next Page ✚

```
0040 ON MARK GOTO @SKIP,@PLUS,   ←If the variable 'MARK' is 0 then go to @SKIP, and if it is 1
                                   then go to @PLUS.
0042 @SKIP
0048 @PLUS
0049 PRINT NO1;"+";NO2;"=";NO1+NO2   ←Display the sum of variables NO1 and NO2.
0050 GOTO @LOOP                      ←Return to @LOOP line.
```

The INPUT command means that you wish a value inputted via the keyboard to be assigned to a variable. The IF in line 35 indicates a conditional operation, meaning that if the '+' symbol was entered on the keyboard, the variable MARK will be assigned a '1'. The subsequent lines contain other conditional operations, for instance, if '-' was entered, it will be assigned a '2', if '/' was entered, it will be assigned a value of '3' and so on.

The ON～GOTO command in line 40 introduces several options into the GOTO operation. So if the variable MARK has the value of '0', it will go to a certain line, while if it has a value of '1' or '2' it will go to others. This means that the program will run differently depending on the key that was pressed.

The GOTO command in line 50 indicates that the program will return to the line marked with the @ LOOP label, meaning that the program will be ready to perform another calculation.

In the version of the BASIC language used in Petit Computer, there is no ELSE command, meaning that you cannot have two different processes occurring in the same line of the program. For more details, please refer to the section on branching instructions on page 59.

●Repeated Operations & Array Variables

FOR～NEXT are the typical commands used to repeat the same operation. An array variable is where a number is assigned to multiple variables, allowing them to be managed in a uniform way. They are often used in programming, and examples are given for reference in pages 106 and 113 of Part 5 of this guide.

In the sample program SAMPLE3, a FOR～NEXT command is used to assign data to the array variable DIM ( ). A range of sounds will be played depending on the key pressed, meaning that the program operates like a basic sequencer.

From sample program 'SAMPLE3'

```
0034 DATA "'","A"                ←Multiple data prepared.
0035 DATA "%"
0038 DIM N$(20)                  ←20 array variables from N$(0)～N$(19) prepared.
0040 KCNT=20                     ←Set number of loops with variable KCNT.
0041 FOR I=0 TO KCNT-1           ←Program loops until the value of variable 「I」 is
                                   between 0 and 19.
0042 READ N$(I)                  ←DATA command used to load one of N$ variables.
0043 NEXT I                      ←Returns to line 41.
```

The READ command in line 42 serves to take the 20 pieces of data between lines 24-35 and store them as variables. In order to store them all as variables, this same READ command needs to be repeated 20 times. This is why the variable 20 appears in line 39, meaning that lines 40-43 are looped and the same command is repeated 20 times.

There are a number of other ways to repeat sequences in programs. Please note how line 50 of SAMPLE2 contains a GOTO LOOP@ command, meaning that it returns to the @LOOP label in line 13, so the program will run again and again. Please note that FOR～NEXT operations, where the number of times the process should be repeated can be designated, are different from the looping that takes place using a GOTO command.

How to Write a Program that Repeats 100 Times

```
FOR I=1 TO 100
OOOOOOOOOOOOO          ←Enter process you wish to be repeated here.
NEXT
```

How to Write a Program that Repeats Forever.

```
@MAIN
OOOOOOOOOOOOO          ←Enter process you wish to be repeated here.
GOTO @MAIN
```

Resetting Memory and Screen

The controls and commands below perform a variety of functions, such as resetting the memory or screen, or pausing operations. These commands can be used within programs, and also in Run Mode. There are times when a program does not run correctly because data from a previous program is interfering. If this should occur, try using the commands or pressing the buttons listed below. (Please refer to Part 4 for more details on screen elements, background display and sprites).

+Control Pad + R Button + START	←Reset screen
CLEAR↵	←Clear memory and variable names
VISIBLE 1,1,1,1,1,1↵	←Display all screen elements
GPAGE 0↵	←Return controlled graphic screen to the upper screen
GCLS 0↵	←Delete graphic screen
BGPAGE 0↵	←Return BG screen to upper screen
BGCLIP 0,0,31,23↵	←Reset BG display range
BGOFS 0,0,0↵	←Reset BG front screen offset
BGOFS 1,0,0↵	←Reset BG rear screen offset
SPPAGE 0↵	←Return controlled sprite screen to upper screen
SPCLR↵	←Delete sprite
BGMSTOP↵	←Stop background music

Using Operators for Calculations and Variables

One of the fundamentals of programming is the repeated process of input, calculation and output. This makes it vital to have a solid grasp of how to assign inputted values to variables, and how arithmetical operators are used to calculate values. Before starting to write your own programs, it's best to review these programming basics.

●Numerical Values

This software uses 32 bit fixed-point numbers, with fractions rounded up. Integers within the range of ±524287 can be used (internally, 4096 is treated as 1.0). Numbers can be written in either hexadecimal or binary form.

Type	Display
hexadecimal	&H
binary	&B

How to program

```
PRINT &H0F↵          ←A hexadecimal value will be displayed as a decimal value. In this case, '15' will be displayed.
PRINT &B11111111↵    ←A binary value will be displayed as a decimal value. In this case, '255' will be displayed.
```

●Calculation and Arithmetical Operators

Arithmetical operators are the signs and symbols used when performing calculations. In Petit Computer, the remainder in a division calculation is represented as '%', which is the same as in the C programming language.

Priority	Calculations
High	( ) []
	MINUS NOT
	Function
	* ／ %
	＋ －
	== != < <= > >=
Low	AND OR XOR

How to program

```
PRINT 10/3↵          ←This division calculation will not give a whole number. 3.333 will be displayed.
PRINT 2+3*3↵         ←In this calculation, the multiplication will take place first, meaning that the result will be '11'.
```

●Bit Operators

Bit operators are also known as logical operators and are the signs and symbols used to perform logical operations. Using them is a way of determining whether something is true or false in a binary calculation, and is extremely useful when manipulating bits.

Type	Bit Operators
Logical product (if A and B are '1' then '1')	AND
Logical sum (if A or B are '1' then '1')	OR
Exclusive OR operation (if only A or B is '1' then '1')	XOR
Negative (if '1' then '0', if '0' then '1')	NOT

How to program

```
A=A AND 15↵          ←The logical product 15 and the variable A will be assigned to A.
A=A OR &H100↵        ←The logical sum of &H100 and the variable A will be assigned to A.
```

●Relational operators

Relational operators are the signs and symbols used for calculations where values are being compared. In Petit Computer, the signs '=' and '< >' are not used to express whether values are equal or not. As with the C programming language, this is expressed using '==' and '!='.

Type	Relational Operator
Value on left is greater than value on right	>
Value on left is smaller than value on right	<
Value on left is greater than or equal to value on right	>=
Value on left is smaller than or equal to value on right	<=
Both values are equal	==
The values are not equal	!=

Note that when brackets are placed around values, it becomes a logical operation. If the calculation gives a true result, this will be expressed as '1' (for YES). On some other version of BASIC, this is expressed as '-1' so please take note of this difference.

How to program

```
PRINT (1<3)↵          ←Results of comparison is true so '1' is displayed.
```

●Variables

Variables can be given names of a maximum of 8 characters. Alphanumeric characters can be used along with underscore '_'. The names should begin with a letter. If you end a variable name with a '$', it will be treated as a character string.

A' and 'A$' can be used as separate variables.

How to program

```
A=0.333↵              ←Assign numerical value 0.333 to variable A
A$="MOJI"↵            ←Assign character value "MOJI" to variable A$
```

●Array Variables

To perform array declarations, the command DIM is used. An array can have a total of up to 32768 elements, in up to two dimensions. The Parentheses () or [] can be used, and indexing begins from 0. If you define the same array twice, a duplicate definition error will occur. To prevent this, ensure that you use a CLEAR command before any new operation.

For examples of programs that use arrays, please refer to the sample program ① in the 100 Line Programming Corner section on page 95.

How to program

```
CLEAR↵                ←Clears variable.
DIM A(10)↵            ←Create A(0)~A(9) array variable.
A(0)=12345↵           ←Assign 12345 to A(0)
```

How to Run Programs

To check that a programming is running as expected, it's a good idea to regularly switch from Edit Mode to Run Mode to test it out.

This section looks at the commands that are chiefly used in Run Mode. They are all commands that those familiar with BASIC will already know.

```
RUN↵
```

To run a program you have created in Edit Mode, enter RUN in Run Mode and press the Enter key. (You can perform the same operation by pressing the START button).

```
CONT↵
```

Use this command to restart a program that you paused by pressing SELECT or using a STOP command in the program.

```
SAVE"FILE"↵
```

This command is used to save files. If you switch the power off, you will lose the program entered in Edit Mode. To avoid this, use this command in Run Mode to save your files.

```
LOAD"FILE"↵
```

This command is used to load files.

```
EXEC"FILE"↵
```

This command loads files and runs them. It combines both the LOAD and RUN commands in one.

```
FILES↵
```

This displays a list of your files. Use this command when you wish to see the files you have saved.

```
LIST [line number]↵
```

This command switches to Edit Mode and displays the program list. If you omit the line number, the program will be displayed, starting from the first line.

```
NEW↵
```

This command will delete the entire program you are currently writing in Edit Mode.

*File names can be a maximum of 8 characters long, using alphanumeric characters and the underscore '_'.
*FILES"LOAD"SAVE"CONT"RUN' each have function keys you can make use of.

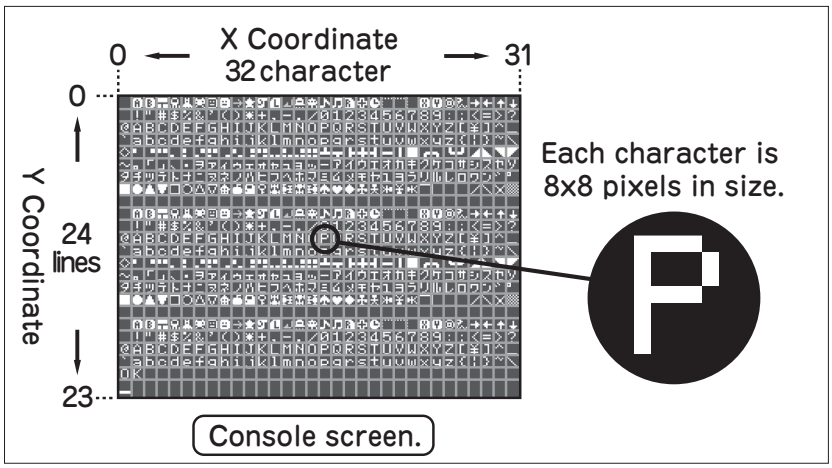
3-02 Displaying Characters

It is now time to actually get programming. Let's begin with an introduction to how to display characters on screen. It's important to grasp how coordinates for the display are used, as well as the particular way the make-up of the display in Petit Computer.

Let's Display Characters

The console screen is where characters are displayed. The console screen appears on the upper screen. Please see page 80 for more information on screen structure. The console screen allows alphanumeric text display of 32 characters x 24 lines.

If you add a new line at the bottom of the screen, the screen will scroll down by a single line.



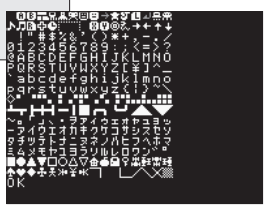
▲32x24 characters can be displayed on the console screen.

Designating Coordinates and Displaying Characters “PRINT,LOCATE”

The PRINT command is a classic BASIC command, and is used to display characters on the console screen. When you want to designate the coordinates where characters should be displayed, you will use a LOCATE command. In this section, we will look at an example program that utilizes the LOCATE command. The desired coordinates are expressed in a formula, and a line break is added every 16 characters.

All characters are displayed on the console screen.(EXAMPLE 3-01)

```
0001 CLS
0002 FOR C=0 TO 255                    ←The process for 0～255 is repeated.
0003 LOCATE C%16,FLOOR(C/16)           ←The display coordinates are set here.
0004 PRINT CHR$(C)                     ←The characters to display appear here.
0005 NEXT
```



▲ Screen when Program is Run(EXAMPLE3-01)

When the program is run, a total of 256 (16x16) characters are displayed on the console screen. The variable C used in the program stands for the character code, and using the function CHR$ prints the characters.

A floor function is one which looks for an integer, so for instance if the argument is '1.234' as in the example program 3-01 displayed above, the value '1' will be returned.

Assigning via the Keyboard(INPUT)

An INPUT command gives alphanumeric characters entered on the keyboard a value and assigns them to a variable. Note that after entering the text, you need to press ENTER.

This is an example program using an INPUT command which analyzes numbers entered via the keyboard.

EXAMPLE 3-02: Distinguishing Between Prime and Non-Prime Numbers

```
0001 CLS
0002 INPUT "スウジ ヲイレテ";A              ←The value entered will be assigned to A.
0003 B=0
0004 FOR D=2 TO A-1                        ←Displayed if value is divisible.
0005 IF (A%D)==0 THEN B=1:? "／";D
0006 NEXT
0007 IF B==0 THEN ?"ソスウ テﾞス"           ←If the value is a prime number.
0008 IF B THEN ?"ソスウ シﾞャナイ"           ←If the value is not a prime number.
```

```
スウジ ヲイレテ?
23651
／67
／353
ソスウ シﾞャナイ
OK
```

▲ Screen when Program is Run(EXAMPLE 3-02)

When the program is run, and a particular value is entered via the keyboard, it will be determined whether or not this value is a prime number. The '?' symbol used in the program stands in for the PRINT command. Try using it in your own programs.

3-03 Control Input Detection

Petit Computer is able to detect which buttons the user presses or which Touch Screen operations the user performs. By making use of this capability, it offers a fun way to program which is only possible on the Nintendo DSi or 3DS systems.

Detecting Button Response

To detect the button response, you use a BUTTON () function. When a particular button is pressed, the function returns a value that turns each bit from '0' to '1'. To see the returned value for each button, please refer to the table to the right. Try using the bit operator AND and write a program which displays whether or not a button has been pressed.

Value Returned for Each Button

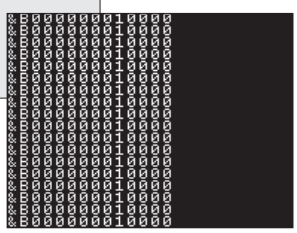
Value Returned	Value Returned (Binary)	Button
1	00000000001	UP
2	00000000010	DOWN
4	00000000100	LEFT
8	00000001000	RIGHT
16	00000010000	A
32	00000100000	B
64	00001000000	X
128	00010000000	Y
256	00100000000	L
512	01000000000	R
1024	10000000000	START

Display the buttons that have been pressed(EXAMPLE3-03)

```
0001 @MAIN
0002 B=BUTTON()                    ←Assign detection of a button state.
0003 M=2048                        ←Reset value used for mask.
0004 PRINT "&B";
0005 FOR I=1 TO 12                 ←Repeat 12 bit section.
0006 PRINT ((B AND M)!=0);         ←Add linebreak after displaying 12 bit section.
0007 M=M/2
0008 NEXT
0009 PRINT
0010 GOTO @MAIN                    ←Display value for a single bit.
```

When the program is run, it will display whether or not buttons have been pressed in binary code over 12 digits. By pressing different buttons, you will be able to see how the values change.

The value returned from SELECT is 2048, but since this button has been assigned to pausing programs, SELECT cannot be detected in programs.



▲The A Button has been pressed in this example.Screen when Program is Run(EXAMPLE 3-03)

Keyboard Detection

If you don't press ENTER after inputting a value, the INPUT command will not be finalized. But using the INKEY$ function will allow detection of the key being pressed at a particular moment. By making use of this function in a program where instantaneous action is called for, you can create something really fun.

The program shown below will enlarge the characters read by the INKEY$ function.

INKEY$ Function (Input by keyboard) (EXAMPLE3-03)

```
0001 CLS
0002 @MAIN
0003 A$=INKEY$()                          ←Single character entry on the keyboard.
0004 IF A$=="" THEN @MAIN
0005 CHRREAD("BGF0",ASC(A$)),BF$          ←Character data input.
0006 FOR I=0 TO 63
0007 V=VAL("&H"+MID$(BF$,I,1))            ←Converts a character string into a numerical value.
0008 IF V==0 THEN PRINT " ";
0009 IF V!=0 THEN PRINT "■";
0010 IF (I%8)==7 THEN PRINT
0011 NEXT
0012 GOTO @MAIN
```

After running the program, any key you touch on the keyboard will be displayed 8 times larger than normal. The function CHRREAD() retrieves character data, while the MID$() function retrieves the designated number of characters from a character string, and VAL() converts them into a numerical value. For more information on functions, please refer to the Petit Computer Resources section at the end of this guide.

You can use this method of enlarging characters in other programs.

▲The characters you enter will be displayed across the whole screen area.Screen when Program is Run(EXAMPLE 3-04)

Touch Screen Detection

Petit Computer makes use of system variables that detect data inputted from the Touch Screen. See the sample program below for an example of this. It deletes the keyboard display on the lower screen and draws on the graphic screen beneath it.

Ink painting by inputted from the Touch Screen. (EXAMPLE3-05)

```
0001 PNLTYPE "OFF"                    ←This removes the keyboard.
0002 GPAGE 1
0003 GFILL 0,0,255,191,247            ←The screen is filled with the color
                                       code 247.
0004 @MAIN
0005 IF TCHST==0 THEN @MAIN           ←The program waits until the screen is touched.
0006 X=TCHX:Y=TCHY                    ←Coordinates are assigned where the
                                       screen was touched.
0007 FOR Y1=-4 TO 4                   ←Repeats for 9x9 pixel section.
0008 FOR X1=-4 TO 4
0009 I=X+X1:J=Y+Y1
0010 C=GSPOIT(I,J)+1                  ←Adds 1 to the color code displayed on screen.
0011 IF C>255 THEN C=255              ←If color code exceeds 255, the
                                       program goes back.
0012 GPSET I,J,C                      ←Draws on screen.
0013 NEXT
0014 NEXT
0015 VSYNC 1                          ←Waits.
0016 GOTO @MAIN
```

If you touch the lower screen after running the program, that section will be colored in. The color will get deeper depending on how long you touch the screen for, meaning that you can create an image that resembles an ink painting. The system variables TCHST, TCHX and TCHY are used to detect the state of the Touch Screen. For more information on system variables, please refer to the Petit Computer Resources section at the end of this guide.

This program also uses the VSYNC command to adjust the speed at which images are drawn. A single frame is around 1/60th of a second. Please also note that once this program is run, graphics will be displayed on the lower screen. Use a GPAGE 0 command to restore this to the upper screen.

▲Screen when Program is Run(EXAMPLE 3-05)

3-04 Drawing Pictures

In the original versions of BASIC, graphic commands tended to differ widely between different systems. The version of BASIC used in Petit Computer is quite different from older version of the programming language.

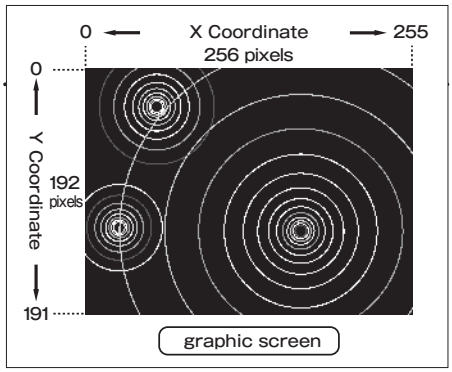
The screen display in Petit Computer is made up of multiple layers. You can program graphics using sprite-based characters and backgrounds. This will be covered in detail in Part 4 of this guide. Prior to that, this section will introduce some of the basics of graphics programming.

The Graphic Screen

There is a graphic screen on both the upper and lower screens, each allowing graphics to be drawn over a range of 256x192 pixels. (Refer to the diagram on the right).

In the version of BASIC used in Petit Computer, functions and commands relating to graphics begin with G～. So for instance, GLINE is the command to draw a line, while GCIRCLE will draw a circle.

Let's begin by drawing lines, dots and circles on the screen.



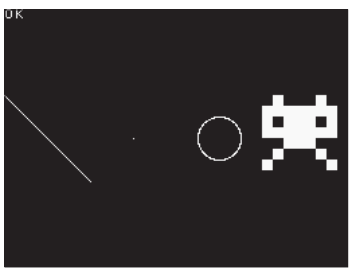
▲The graphic screen has a resolution of 256x192 pixels.

Draw lines, dots, circles and enlarged characters.(EXAMPLE3-06)

```
0001 CLS:GCLS                          ←clear the graphic screen.
0002 GLINE 0,64,64,128,15              ←Draw a line.
0003 GPSET 96,96,15                    ←Draw a point.
0004 GCIRCLE 160,96,16,15              ←Draw a circle.
0005 GPUTCHR 192,64,"BGF0",6,0,8       ←Display characters on the graphic screen.
```

When the program is run, lines, dots, circles and enlarged characters will be drawn on the graphic screen. Unlike on the console screen, the graphic screen gives you the ability to control the graphical display to the level of individual pixels.

Running GCLS ↵ will clear the graphic screen.



▲Screen when Program is Run(EXAMPLE 3-06)

Mixing Colors

The graphic screen can display 256 colors. The following program takes red, green and blue and combines 6 shades of each of these colors, totaling 6x6x6=216 colors.

EXAMPLE 3-07: Display Mix of Red-Green-Blue

```
0001 GCLS
0002 R=3:G=3:B=3                          ←Settings for red, green and blue.
0003 @MAIN
0004 CLS
0005 PRINT "R=";R;" G=";G;" B=";B         ←Displays values for red, green and blue.
0006 C=(R*36)+(G*6)+B                     ←Calculates the color number.
0007 IF C THEN C=247-C                    ←Revises the color number.
0008 GFILL 0,8,255,191,C                  ←Draws a filled square graphic.
0009 VSYNC 6                              ←Waits for a certain period.
0010 @KWAIT
0011 K=BUTTON()                           ←Input via the buttons.
0012 IF K==0  THEN @KWAIT                 ←Waits for button input.
0013 IF K==16 THEN R=(R+1)%6              ←Cycle through shades of red.
0014 IF K==32 THEN G=(G+1)%6              ←Cycle through shades of green.
0015 IF K==64 THEN B=(B+1)%6              ←Cycle through shades of blue.
0016 GOTO @MAIN
```

When the program is run, it will display a combination of these colors, depending on the values entered for red, green and blue (R, G, B). The A Button is assigned to red, the B Button is assigned to green, while the X Button is assigned to blue. By pressing each of these buttons, the value (0-5) for these colors will change. So, for example, by increasing the values assigned to green and blue, you can create a particular watery shade of blue.

Branching

To use a command where you want the action taken to differ depending on conditions, use either an IF～THEN or IF～GOTO command. This type of command is essential in BASIC programming. With other versions of BASIC, an ELSE command is used after the branch instruction telling the program what to do if a certain condition is not met, or an ENDIF command is used to end the conditional processing section of the program. In the version of BASIC used in Petit Computer, ELSE and ENDIF commands cannot be used. To carry out a branching operation, you use a combination of the IF～THEN and GOTO commands. Please see the example program in this section to see these commands in action.

How to program

```
IF (   ) THEN @YES
OOOOOOOOOOOOO  ←Enter the process to take place in the event of a NO.
GOTO @GORYU
@YES
OOOOOOOOOOOOO  ←Enter the process to take place in the event of a YES.
@GORYU
OOOOOOOOOOOOO  ←Here, the program continues whether it was YES or NO.
```

3-05 Playing Sounds

As we come to the end of Part 3, we will look at how to perform audio programming using Petit Computer. The software comes with a rich variety of audio content in its sound library. In this section, we will look at some programs that make use of these sounds.

Playing Background Music

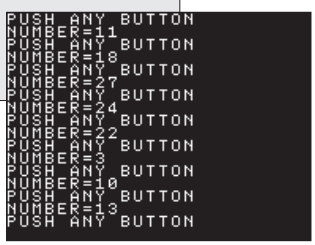
There are 30 background music tracks that come pre-loaded on Petit Computer. By using the BGMPLAY command, you can play a piece of music on a loop while a program is running.

Playing Background Music on a Loop(EXAMPLE3-08)

```
0001 CLS
0002 @MAIN
0003 PRINT "PUSH ANY BUTTON"
0004 @WAIT
0005 IF BUTTON()==0 THEN @WAIT   ←The program waits for button input.
0006 N=RND(30)                   ←A random number between 0-29 is generated.
0007 PRINT "NUMBER=";N
0008 BGMPLAY N                   ←A background music track is played
0009 VSYNC 30                    ←Pause for 0.5 seconds.
0010 GOTO @MAIN
```

When the program is run, pressing any button (with the exception of SELECT) will play a random music track from the 30 included with the software. If you cannot hear anything, please check the system volume.

Please note that even if you stop this program, the music will continue playing. To stop the music, you will need to enter a BGMSTOP ↵ command in Run Mode.



▲The screen display when example program 3-08 is run. Press any button and a music track will be played at random, and the BGM track number will be displayed.

Playing Sound Effects with the BEEP Command

You can play up to 70 types of sound effect using Petit Computer. To play a sound effect, you use the BEEP command. In example program 3-09, a BEEP effect is played whenever a button is pressed.

Playing the BEEP.(EXAMPLE3-09)

```
0001 CLS
0002 @MAIN
0003 PRINT "PUSH ANY BUTTON"
0004 @WAIT
0005 IF BUTTON()==0 THEN @WAIT   ←The program waits for button input.
0006 N=RND(70)                   ←A random number between 0-69 is generated.
0007 PRINT "NUMBER=";N
0008 BEEP N                      ←A sound effect is played.
0009 VSYNC 30                    ←Pause for 0.5 seconds.
0010 GOTO @MAIN
```

▲The screen display when example program 3-09 is run. Press any button and a sound effect will be played at random, and the sound effect number will be displayed.

When the program is run, pressing any button (with the exception of SELECT) will randomly select a sound effect from the selection of 70 and play it. You never know what sound effect will be played, giving this program an enjoyable unpredictability.

Playing a Melody Using Sound Effects

The BEEP command can do more than simply play regular sound effects. By adjusting the pitch of the sound effects, you can play them in different octaves. The example program below shows how this function can be used.

When the program is run, a piano effect will play 'Do Re Mi Fa So La Ti Do' in a continuous loop. The pitch will shift relative to the last tone, so it is not an absolutely accurate reproduction of piano scales. The values in the DATA line in the program represent musical intervals or steps. The variable V represents the volume, the variable W represents the tone, while the variable P controls the panning of the audio between the right and left speakers. Try using the techniques used here in your own programs.

Playing Do Re Mi repeatedly(EXAMPLE3-10)

```
0001 W=22 'WAVE(0-69)
0002 V=80 'VOLUME(0-127)
0003 P=64 'PANPOT(0/64/127)
0004 @START
0005 RESTORE @SCORE
0006 @MAIN
0007 READ I
0008 IF I>=99 THEN @START
0009 I=I*4096/12
0010 BEEP W,I,V,P
0011 VSYNC 15
0012 GOTO @MAIN
0013 @SCORE
0014 DATA 0,2,4,5,7,9,11,12,99
```

MML can be played too.

In this section, we will look at MML, which is also introduced on the official Petit Computer website. MML is a well-known form of syntax used to convert music into the form of data. This program decodes this syntax and replicates this kind of music using only BEEP commands. This will give you a taste of how it feels to make music using a MIDI device.

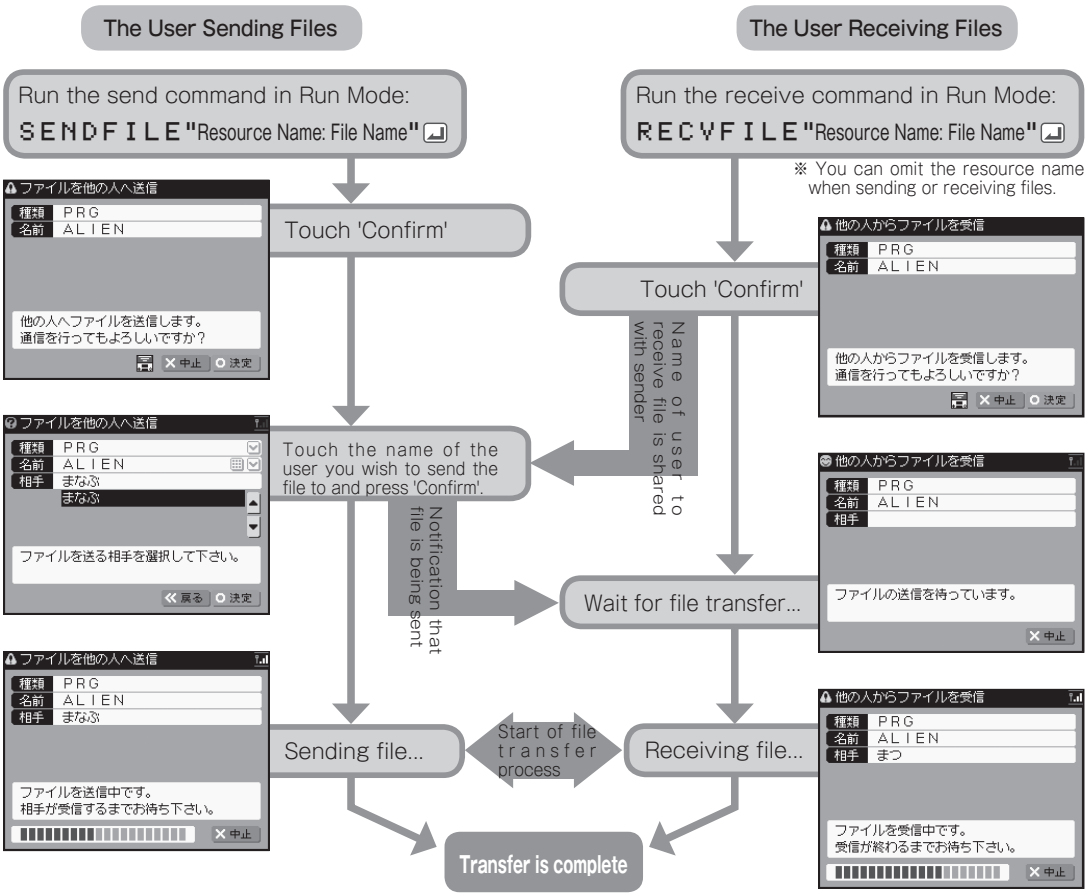
◀ MML data is being played. The 8 instruments are layered to create the audio track.

3-06 Exchanging Files With Other DSi and 3DS Users

On Petit Computer, there is no function which enables you to upload files saved on your Nintendo DSi or 3DS system to a computer. This means that you cannot save programs to an external device, or distribute your program data to a large number of people. However, you can exchange program files with other DSi and 3DS users one-to-one via Local Wireless Communications. The more fellow Petit Computer users you have around you, the more fun you'll have.

Sending and Receiving Files

If you have two Nintendo DSi or 3DS systems with Petit Computer, you can send and receive files via local wireless communication. The guide below shows you how.



When you receive a file, you don't have to use the same file name that the sender used. You are free to modify it to a name of your choice.

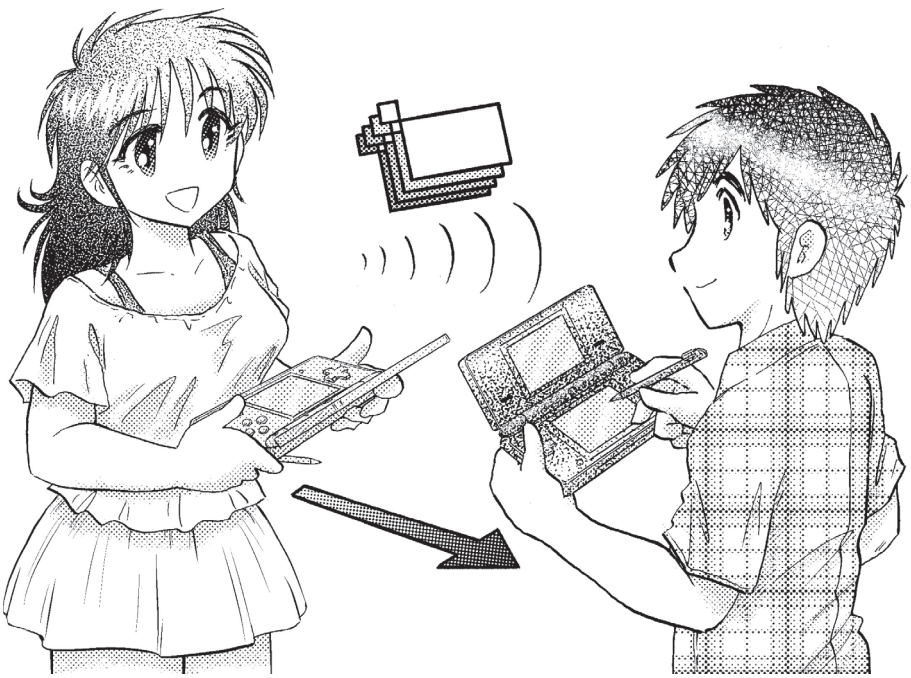


Image: Hiroshi Kuri

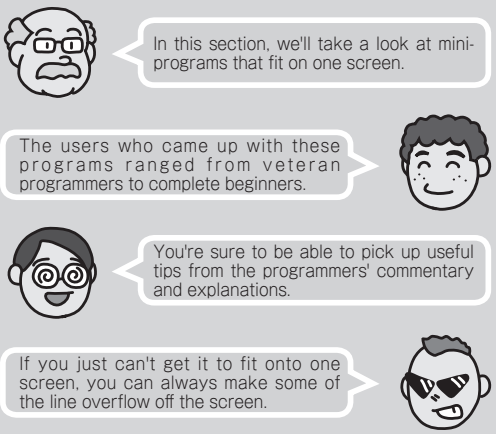
You will have noticed from the guide that the resource type is displayed on the sender's screen. See Part 4-01 for more information on this. In addition to program files, Petit Computer characters, screen data and memory data files can also be transferred between users.

Special Section

Byte-sized Programs to Enjoy

Single-Screen Programming Corner

For readers of this guide who are familiar with BASIC to some degree but have not used it for a long time, we hope that the contents thus far have inspired memories of programming, and that you have got to grips with the fundamentals of Petit Computer. At this point, we would like to introduce you to a few small programs which the author and associates came up with. All of these programs fit on a single screen of Petit Computer, which is 29 characters x 24 lines in Edit Mode. Test your programming skills and see what you can achieve with a program in such a small space.



Entry No.1

Okinawan Traditional Folk Music Program

RYUKYUTRONICS

developer : tac

● Introducing the Program

Give the Touch Screen a few taps with the stylus, or draw a nice big circle for a fantasia of traditional Okinawan folk.

▲Touch the lower screen and the Sound is played. It is heard like melody by moving.

● More About the Programmer

I was born in 1966 and learned BASIC as a high school student on the Commodore VIC-1001 and the single-board TK-80BS. I also dabbled in assembly language. Subsequently, I made sound effects for TV, films and games and now make websites. I am not a programmer by profession, but it will always be part of my life.

●How it Works

Slide the stylus across the Touch Screen in any way you choose. The center of the screen forms the starting point, with left and right shifting the sound over 2 octaves. The y-axis is not visible. The characters "ドんやヨレ" in the second line represent the musical scale typical of traditional Okinawan folk music (after being converted to ASCII, they are assigned to different notes in the scale). The fact that you cannot program a conditional statement in the FOR~NEXT line, with more than one command in a single line is a bug and created a bit of a challenge here... But this has been fixed in the updated version 1.1 of the software. Though it is rather lacking in visual style, with only the cursor appearing, it more than makes up for it with its interesting approach to sonics.

Program List

```
0001 PNLTYPE "OFF":CLEAR:CLS
0002 DIM D(21),L(50):J$="ドんやヨレ"          ←The variable J$ is used to create the musical scale.
0003 SPPAGE 1:SPSET 0,112,0,0,0,0
0004 FOR I=0 TO 3                              ┐
0005 FOR II=0 TO 4                             │
0006 B=ASC(MID$(J$, II,1))                     ├ The variable J$ is used to create the musical scale.
0007 D(II+1+(I*5))=B+(I*12)                    │
0008 NEXT II                                   │
0009 NEXT I                                    ┘
0010 FOR E=0 TO 49:L(E)=-1:NEXT E
0011 @START
0012 X=TCHX:Y=TCHY:SPOFS 0,X-8,Y-8            ←Displays characters at location on screen that is touched.
0013 SC=FLOOR(X/12.8)+1:O=D(SC)                ┐ The coordinate on the x-axis where the screen is
0014 S=(-8192+((8192/24)*O))                   ┘ touched is assigned to the variable S, and a sound is played.
0015 L(0)=-1
0016 IF TCHST==TRUE THEN GOSUB @SP             ←A sound is played when the screen is touched.
0017 FOR E=1 TO 49                             ┐
0018 IF L(E) !=-1 THEN GOSUB @SP2              ├ This is the echo effect process.
0019 L(50-E)=L(49-E):NEXT E                    ┘
0020 VSYNC 3:GOTO @START
0021 @SP
0022 BEEP 16,S:L(0)=S:RETURN                   ←The subroutine that plays sound.
0023 @SP2
0024 BEEP 16,L(E),(49-E)*2:RETURN              ←The subroutine that replays sound for an echo effect.
```

Check Point

The tone is that of a traditional Japanese string instrument. There may be no images, but it sounds great!

To stop the music getting monotonous, this program uses echo effects and changes in pitch depending on where the screen is touched.

The echo effect can play an array of up to 50 sound effects, so it is possible that the sound effects will be slightly delayed. If this occurs, try reducing the number of sound effects in the array, and try adding a VSYNC command between rows 17-19.

Entry No.2

Feel an Exciting Rhythm With Your Fingertips

DDREV

developer : tac

●Introducing the Program

This program lets you enjoy a simple rhythm game. The aim is to press the buttons in perfect sync with the rhythm. You're sure to find it quite a challenge.

●More About the Programmer

Well, I'm also responsible for Entry No.1. I hadn't programmed with BASIC for a long time, so I got so excited that I came up with two programs in a row.

●How it Works

When the musical notes reach the first line, at the top of the screen, you'll need to press either the A or B Button. Press the B Button when the note is on the left, and press the A Button when it is on the right. Match the rhythm and aim for a high score. When you are perfectly in sync, you'll hear a dog or cat noise. When you miss a beat, you'll lose points.

The important point from a programming perspective is the use of the IF line for the rhythm. This program was created at our office, and the cat and dog noises ended up really annoying everyone sitting nearby.

▲Two columns of notes move up the screen. Press the A or B Button as they reach the top of the screen.

Program List

```
0001 CLEAR:CLS:R=68:P=69                    ←The sound effect codes are assigned to variables R and P
0002 @START
0003 GCLS 0
0004 IF (C%4)==0 THEN BEEP 31,0,99          ┐
0005 IF (C%8)==4 THEN BEEP 30,0,99          ├ The rhythm is created with a combination of 3 instruments"
0006 BEEP 27,0,64                           ┘
0007 C=C+1:IF C>15 THEN C=0
0008 K=BUTTON():LOCATE 0,0:?"¥:";S
0009 IF K==16 THEN D=E                      ←The process when the A Button is pressed.
0010 IF K==32 THEN N=I                      ←The process when the B Button is pressed.
0011 S=S+D+N:Z=D+N:D=D*7:N=N*7              ←The calculation when you hit the beat and score points.
0012 IF(K!=0 AND Z==0)THEN GOSUB@E          ←The subroutine that is initiated when you miss a beat.
0013 BEEP R,0,D:D=0:BEEP P,0,N:N=0
0014 VSYNC 10:GOSUB @SCORE
0015 I=CHKCHR(20,0):E=CHKCHR(24,0)
0016 GOTO @START
0017 @SCORE
0018 X=RND(5):Y=(RND(2)*4)+20               ←The note coordinates → variable Y, display flag → variable X
0019 IF (C%2)!=0 THEN X=9
0020 IF X==0 THEN LOCATE Y,23:?"♪"          ←This displays a note on the last line and makes it scroll.
0021 IF X!=0 THEN LOCATE 0,23:?" "
0022 RETURN                                 ←The process when the player misses a beat.
0023 @E
0024 BEEP 0:S=S-16:GCLS 100:RETURN
```

Check Point

At first, I didn't have a clue which button to press and when.

You need to pay attention to lines 4-7, which deal with the rhythm. It's a combination of one instrument playing 16 times, another one playing 4 times, and another one playing twice.

Entry No.3

A Simple Action Game with Sprites and a Scrolling Background

One Key Jump

developer : Hiroshi Yamasaki

● Introducing the Program

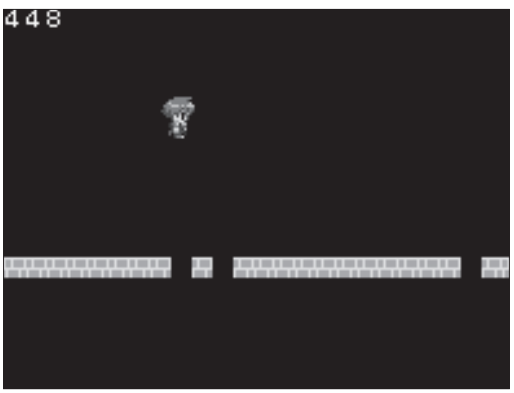
The screen scrolls automatically from right to left, meaning you have to time your jumps correctly to clear the holes in the ground. The holes appear at random, so the difficulty varies, but there is bound to come a time when you find it too hard to handle.

● More About the Programmer

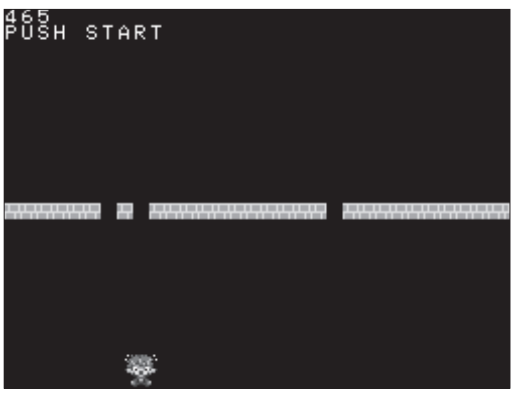
I've been involved with computers for 27 years, since first learning BASIC on the Sord M5. After learning machine code and getting programming experience on the M5 and MSX2, I wrote for the Micon BASIC Magazine, introducing the readers to BASIC on the Sega Saturn and PlayStation 2, and writing articles about all aspects of programming. I still program to this day.

● How it Works

When the program is run, it will wait for button input before the game begins. Press START and the action will get underway. Press any button to jump, and aim to clear the holes in the ground. Fall into one and it's game over. Your score is displayed in the top left, representing the distance you have covered. Press START at any point to restart the game.



▲The screen scrolls from right to left.



▲Mistime your jump and you will fall down one of the holes, and it will be game over.

Program List

```
0001 @1
0002 PRINT "PUSH START"
0003 FOR L=0 TO 1024:L=BUTTON()
0004 NEXT:BGMPLAY 14:SPCLR:CLS          ←The FOR line means that the program will wait for START to be pressed.
0005 CLEAR:SPSET 0,64,2,0,0,0
0006 SPANIM 0,4,6:FOR I=0 TO 63
0007 BGPUT 0,I,0,96,2,0,0:NEXT          ←The game's start point. The ground will be filled in.
0008 FOR L=0 TO 2:VSYNC 1               ←The main loop.
0009 A=0:IF RND(99)>1 THEN A=1          ←Generates a random number. If the variable A = 1, the ground will remain the same.
0010 BGPUT 0,X/8-1,A,0,2,0,0            ←If variable A = 0, a hole in the ground will be generated.
0011 F=0:IF Y<0 OR J!=0 THEN F=1        ←If the character is in mid-air or mid-jump, the value 1 will be assigned to variable F.
0012 IF BUTTON()==0 THEN F=1            ←If no button is pressed, the value 1 will be assigned to variable F.
0013 IF F==0 THEN J=-8:BEEP 8           ←If the variable F is 0, the character will begin his jump.
0014 SPOFS 0,60,Y+80:BGOFS 0,X,416
0015 BGREAD(0,X/8+8,0),C1,H,P,V         ┐ The data relating to whether or not there is a
0016 BGREAD(0,(X+7)/8+8,0),C,H,P,V      ┘ hole in the ground will be assigned to variable C. Calculation for jump and drop.
0017 C=C1 OR C
0018 Y=Y+J:J=J+(Y<0)*0.5-(Y>=0)*J       ┐ The score (X coordinate of scrolling screen) is added
0019 X=X+1:LOCATE 0,0:PRINT X           ┘ up and displayed.
0020 IF C==96 OR Y<0 THEN L=0:NEXT      ←If there is no hole in the ground, or the character is mid-jump, this section will loop.
0021 BGMSTOP:BEEP 6:FOR J=0 TO 13       ┐
0022 Y=Y+J:SPOFS 0,60,Y+80:VSYNC 1      │ The Game Over process.
0023 NEXT:SPCHR 0,88,2,0,0,0            │
0024 GOTO @1                            ┘
```

Check Point

This is impressive! It's a short program, but it still finds space for score-keeping and a game over process.

The collision detection covers the lower central part of the player character. If there are 8 pixels open beneath him, he will fall and it's game over.

The ground will become more and more riddled with holes, so in the end, you won't be able to get across, no matter how much you jump. If you modified this program to allow the player to move right and left, it could be a lot of fun.

Entry No.4

A Game for Killing Time

Puchipuchi

developer : Manabu

●Introducing the Program

When you've got time to kill, there are few things more satisfying than popping bubble wrap. This program was created with that same feeling in mind. The reason the same image appears on both the upper and lower screen is that the programmer wasn't sure how to detect the coordinates of the Ë on the Touch Screen.

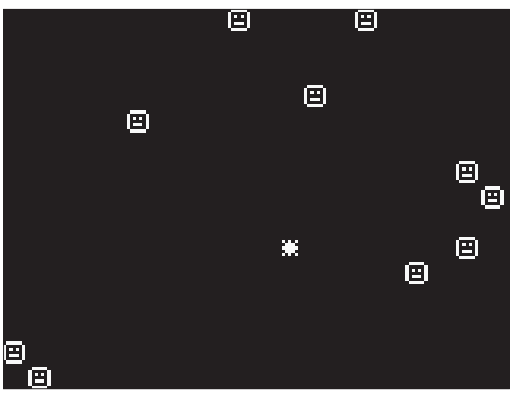
●More About the Programmer

I was born in 1965 which puts me right in the middle of the BASIC generation. At the time, I was content playing games created by other people, though I did give N-BASIC on the PC-8001 a try, but gave up when I came up against arrays. I attended some lectures on FORTRAN at university, but I still found arrays to be problematic and failed to get credit for the course. And that's where I stand today.

●How it Works

There are many "😐" on the touch screen, and touch by the pen and delete them one by one.

After clear, you can see the very little event(only one message is displayed).



▲You'll find lots of smiley faces on the Touch Screen. Touch them with the stylus and they will pop and disappear.

▲When you make them all disappear, a message will be displayed.

Program List

```
0001 CLS:GCLS:CLEAR:PNLTYPE "OFF"
0002 GPAGE 1:M=30:DIM X(M),Y(M)          ←Assigns the number of symbols to variable M.
0003 Z=0:FOR I=0 TO M-1
0004 @RE
0005 X(I)=RND(31):Y(I)=RND(23)           ←Requests the coordinates for the symbol display.
0006 C=CHKCHR(X(I),Y(I))                 ←If there is already a symbol in the same location, it will
                                          start the process again.
0007 IF C==7 GOTO @RE
0008 PNLSTR X(I),Y(I),"☺",0              ←Displays characters on the lower screen.
0009 LOCATE X(I),Y(I):?"☺":NEXT I        ←Displays characters on the upper screen.
0010 @TOUCH
0011 V=FLOOR(TCHX/256*32)                ┐ Detects the coordinates of place screen was
0012 W=FLOOR(TCHY/192*24)                ┘ touched.
0013 IF TCHST==0 GOTO @TOUCH             ←Waits until the screen is touched.
0014 IF CHKCHR(V,W)!=7 GOTO @TOUCH       ←If the screen is not touched, it will return to an earlier
                                          line.
0015 Z=Z+1:FOR I=0 TO 2
0016 PNLSTR V,W,"*",0
0017 LOCATE V,W:PRINT "*":VSYNC 3
0018 PNLSTR V,W," ",0
0019 LOCATE V,W:PRINT " ":VSYNC 3
0020 NEXT I
0021 IF Z!=M GOTO @TOUCH                 ←If all symbols have not been erased, it will return to an
                                          earlier line.
0022 @OWARI
0023 LOCATE 9,9
0024 PRINT "♪CLEAR&CLEAN!♪"
```

Check Point

Hey! I want to hear a nice popping sound when I touch the smiley faces!

Well, why don't you add it yourself? Maybe you could add a BEEP command between rows 14 and 15.

This program uses array variables for X and Y coordinates, but I don't really think this is required. All you need is variables for X and Y.

Entry No.5

A Sound-Effect Sampler

Puchimin

developer : D Kumayaro

● Introducing the Program

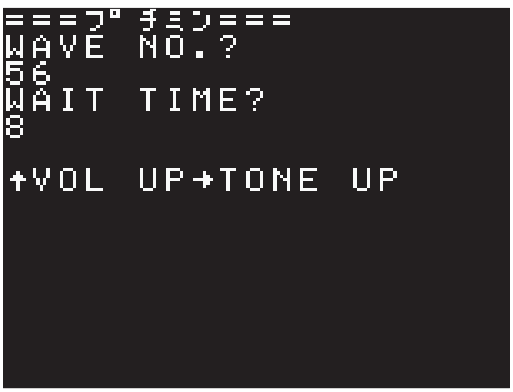
This program resembles the first single-screen sample program we showed you, but the calculation process and the way it plays the sound effects are a little unpolished. It took about two days to create, including time spent learning how to use the software. The original idea was to mimic a Theremin, the electronic instrument that changes pitch as you move your hand closer to it. However, it ended up being something completely different. If the Touch Screen allowed detection in two different places at the same time, this program could have been designed to use two styluses, but alas, it wasn't meant to be...

● More About the Programmer

I am 42 years old, and it's been 15 years since I last programmed with BASIC. I've loved fiddling around with machines since I was a child. I was really struck by the NEC single-board TK-80 and I learned BASIC on the NEC PC-8001/8801. My friends and I used to go to the local electric goods store and write silly programs on the computers there. I now work in sales in the IT industry, though my knowledge of the field remains stuck in the past. Still, I do my best to pretend I know what I'm talking about...

● How it Works

After the program starts, the code for the BEEP sound effect and interval between sound effects is set. When you slide the stylus along the Touch Screen, it plays the sound effect. If you slide the stylus upwards, the volume increases, while if you slide it to the right, it rises in tone. If you make circular motions with the stylus, it has a kind of vibrato effect.



▲ If you select a percussion-style sound effect, and set a suitable interval between effects, you can create a great beat.

▲ Sliding the stylus from right to left will alter the pitch of the note, while sliding it from top to bottom will change the volume.

Program List

```
0001 CLS:CLEAR:PNLTYPE "OFF"
0002 PRINT "=== ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ==="
0003 INPUT "WAVE NO.";WAVE          ←Waveform number input (0-69)
0004 INPUT "WAIT TIME";WTIME        ←Time period input (1-)
0005 PRINT ""
0006 PRINT " ↑VOL UP →TONE UP"
0007 @START
0008 IF TCHST==TRUE THEN GOTO @CAL  ←Creates sound effect when screen is touched.
0009 GOTO @START
0010 @CAL
0011 TONEX=(TCHX*64)-8192           ←Calculates the sound's pitch.
0012 VOLY1=FLOOR(TCHY*127/192)      ┐
0013 VOLY2=ABS(VOLY1-127)           ┘ Calculates the volume's pitch.
0014 BEEP WAVE,TONEX,VOLY2          ←Plays sound effect.
0015 VSYNC WTIME
0016 GOTO @START
```

Check Point

So will this program let you use the Touch Screen to shift the pitch of a sound effect up or down by a range of two full octaves?

Well, it's a bit complex, but the maximum value of the pitch formula on line 11 cannot reach 8192 (2 octaves). But perhaps you could try adjusting it as follows: **TONEX=((TCHX/255)*16384) - 8192**. What do you think, Prof?

Hmmm... I'm afraid that's still not going to do it. It's tricky to touch the corners of the screen with the stylus, so it's hard for the returned value of TCHX to ever reach 255. It's a good idea to adjust the processing for the upper limit on inputted values to allow for larger values.

Entry No.6

A Program That's Like Watching Grass Grow

Lawn Ranger

developer : Matsubara

●Introducing the Program

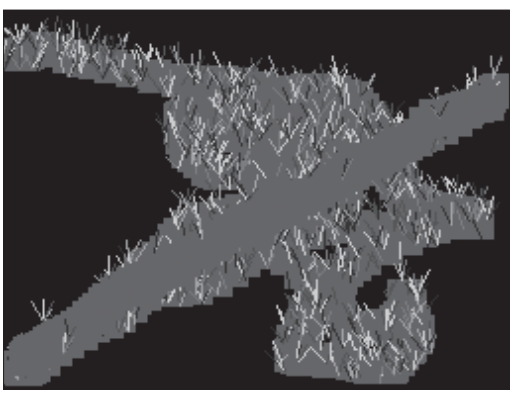
This environmentally-sound program lets you watch grass grow on the screen.

●More About the Programmer

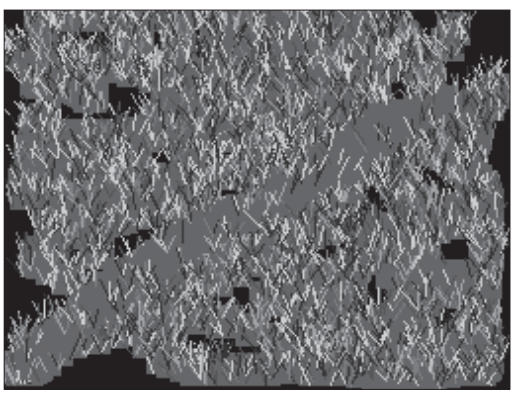
Many years ago, when I was a student, I had my programs published in a BASIC magazine. Compared to the trouble I had programming with the 8 colors available on the Sharp X1, Petit Computer was a breeze. There are all sorts of shades of green, for instance, which really impressed me. I think this comes across in my program.

●How it Works

Press the A Button to select 'Earth', the B Button to select 'Water', or the X Button to select 'Concrete'. Touch the lower screen to place the selected item on the screen. When you place earth on the screen, grass will grow, letting you sculpt your own lawn.



▲Position earth on the Touch Screen and grass will grow from it.



▲Wait a while and the grass will cover the screen. There are no particular surprises in store beyond that.

Program List

```
0001 CLS:PNLTYPE "OFF":BGMPLAY 20
0002 CA=7:CB=4:CC=13:C=CA                    ←Settings for the color code.
0003 GPAGE 1:GCLS
0004 @MAIN
0005 VSYNC 1
0006 X=RND(256):Y=RND(192)
0007 N=0:CX=GSPOIT(X,Y)                      ←Obtains color code with random coordinates.
0008 IF CX==CA THEN N=RND(4)+1               ←If earth has been placed on the screen, grass will grow.
0009 FOR I=1 TO N                            ←Variable N represents the number of blades of grass.
0010 T=RAD(225+RND(90))                      ←Variable T represents the angle at which the grass grows.
0011 R=RND(6)+6                              ←Variable R represents the length of the grass.
0012 CX=241-(RND(5)*6)                       ←Variable CX represents the color of the grass (5 shades
                                               of green).
0013 X2=X+COS(T)*R:Y2=Y+SIN(T)*R
0014 GLINE X,Y,X2,Y2,CX                      ←Draws grass.
0015 NEXT
0016 B=BUTTON():IF B THEN BEEP 5
0017 IF B==16 THEN C=CA                      ←Selects earth.
0018 IF B==32 THEN C=CB                      ←Selects water.
0019 IF B==64 THEN C=CC                      ←Selects concrete.
0020 IF TCHST==0 THEN @MAIN
0021 X=TCHX:Y=TCHY:BEEP 30
0022 X1=X-8:X2=X+8:Y1=Y-8:Y2=Y+8
0023 GFILL X1,Y1,X2,Y2,C                     ←Draws image on screen.
0024 GOTO @MAIN
```

Check Point

Um... So what exactly is the point of this program?

Hmmm... It would have been nice to have a few more game elements, such as more grass growing near water.

This reminds me - someone was working on a program that filled the screen with fingerprints, but they ended up abandoning it.

Extra!

~ Petit Computer Programming Tips ~

A Smart Approach to Programming

Let's begin by identifying some particular problems you may encounter when programming with Petit Computer.

- You weren't really sure what to write, and a line in your program has got really long. You have ended up confused as to what this or that particular part of the line does.
- The order that operations and processes take place has become really confused. You have come up with what you might call 'spaghetti code'.
- The characters have extended to the right edge of the screen, and they're hard to read. Each line in Petit Computer is 32 characters long. When I exceed that line length, I have to scroll character by character to read more.

Here we will introduce you to some special methods for writing programs in order to avoid problems like this.

● Abbreviation and Omission of Characters

It's good to keep lines as short as possible so that they are easy to read on screen. This will mean you won't have to scroll sideways to read programs.

How to program

```
?"ABC"                    ←Abbreviating PRINT commands to ?
IF A THEN @MAIN           ←Omitting GOTO
```

● Use Subroutines

Divide programs up into manageable segments. Rather than writing long lines with very intricate processes, create separate sections that carry out the desired operations. You can then use GOSUB commands to go straight to these subroutines, making the whole way the program operates smoother and easier to grasp.

How to program

```
GOSUB @SUB1
GOSUB @SUB2
```

● Avoid Using GOTO Wherever Possible①

On Petit Computer, you cannot use ELSE commands after IF ~ THEN, so try this alternative.

How to program

```
IF A==0 THEN GOSUB @SUB1
IF A==1 THEN GOSUB @SUB2
```

●Avoid Using GOTO Wherever Possible①

Depending on the values you input when using conditional branching in your program, it may be easiest to use ON～GOSUB commands to fit the operation on a single line. This will keep things shorter than using an IF～GOTO command and listing the different operations, and will make the program clearer to read.

How to program

```
ON A GOSUB @SUB1,@SUB2          ←If value of A is 0, go to SUB1. If it is 1, go to SUB2.
```

●Leave Notes

Make notes in each section of the program so that you can look back and see clearly what each line does.

How to program

```
X=0   '----Xｻﾎﾟﾄｮｳ          ←The text after the apostrophe is a comment, and will not affect
                                the operation of the program itself.
```

There are ther ways of preventing lines in your program from getting too long and requiring you to scroll. These include avoiding long variable names, and avoiding the use of indentation. This may seem out of step with the way things are done nowadays, but it's a good way of keeping your programs in Petit Computer nice and clear.

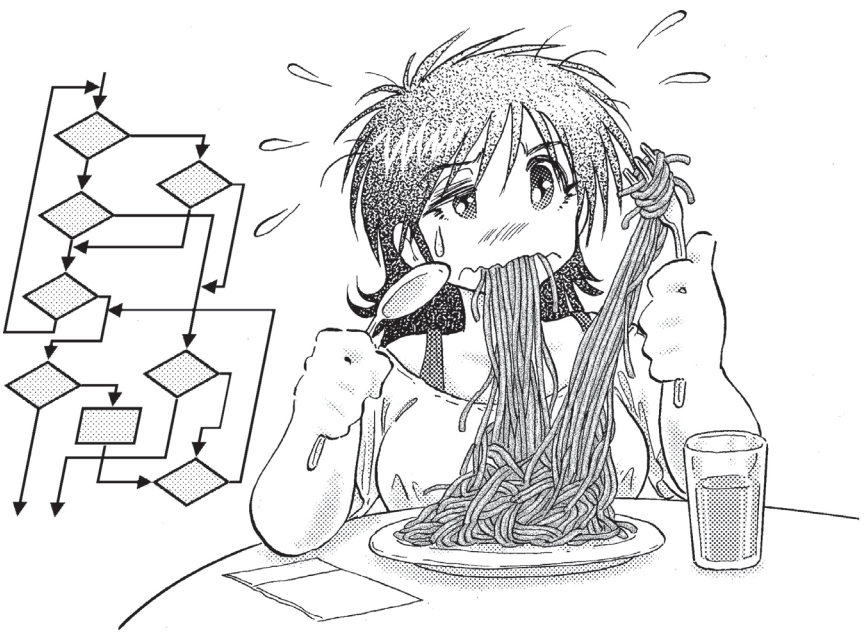


Image:Kuri Hiroshi

Stars of BASIC④

Dento Teramachi

I was the foolish kid cutting class on the 8th of every month so he could read the latest issue of my favorite BASIC magazine in the book store.

My friends and I whiled away our youths listening to our favorite singers and trying to cut out all the IF codes we could. I remember watching agog as someone put together a routine using about six sets of brackets, and managed to get an airship moving in eight directions with a single line of code.

'How on earth did you manage that?'

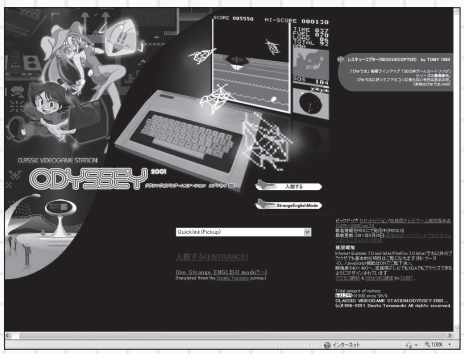
'I just gave it a try, and came up with this.'

So it was BASIC that taught me the true meaning of the word envy.

Dento Teramachi runs a website called Classic Videogame Station Odyssey (http://www.ne.jp/asahi/cvs/odyssey/).

Looking at the site, the links between the latest games machines like the Nintendo 3DS and old versions of BASIC can be traced.

Working under the pen name Fu-San, Teramachi was an illustrator and contributed popular programs to 'Micom BASIC Magazine'.



▲The Classic Videogame Station Odyssey homepage. It is a treasure trove for fans of vintage gaming.

Stars of BASIC⑤

Yukihiko Tani(Bug Taro)

During the first home computing boom, I was in my second year of junior high school. I remember visiting the electronic department in the big stores and being able to get my hands on the display models from all the main manufacturers. For someone like me who loved computers but had no hope of affording one, it was a dream come true.

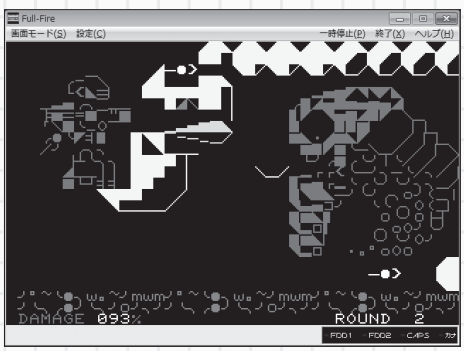
At the time, BASIC would start when you booted up most computers.

This meant that almost everyone had a working knowledge of BASIC.

I remember my friends and I would all imagine that we could master BASIC and then port across any arcade game we liked to our home computers. We were being a little over-optimistic there...

Now, with Petit Computer I think you really could port the arcade games of the 80s across!

Yukihiko Tani began programming when he was in junior high school and his programs subsequently appeared in My Con BASIC Magazine (Dempa Shinbunsha) under his pen-name Bug-Taro. His programs pushed the technical boundaries of the time and proved to be exceptionally popular with readers. He went on to work for a game development company and has been involved in creating a huge number of games. He is now the CEO of Next Entertainment Co.

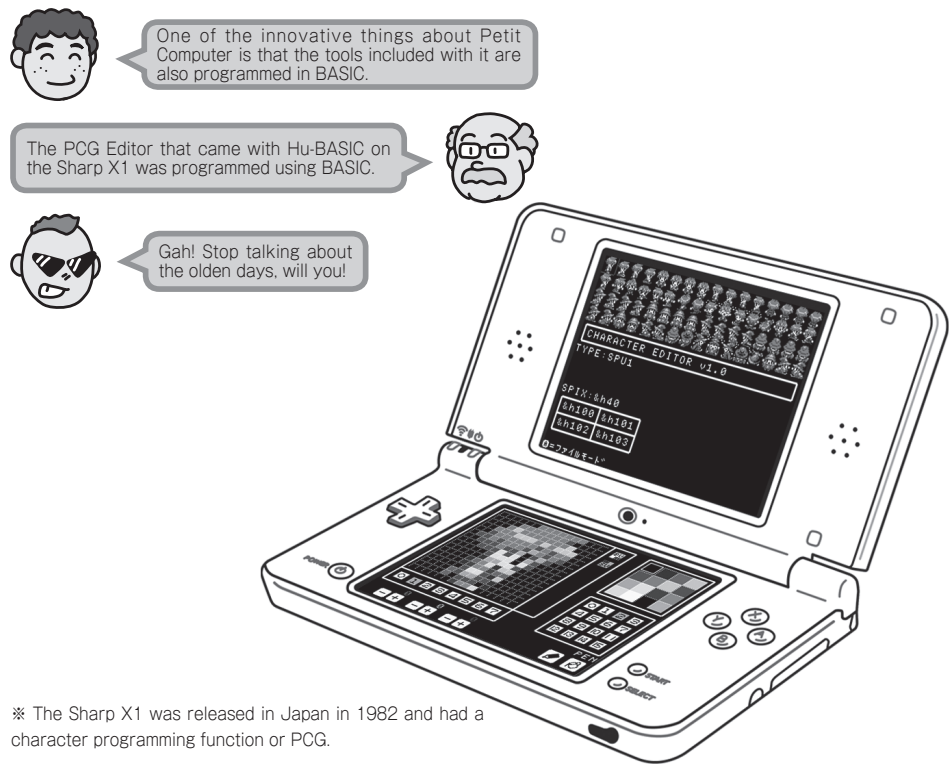


▲Full-Fire, a program that appeared in October 1990. The image is taken from a publication on the NEC 8-bit PC-8001 and PC-6001.

PART 4

Creating the Graphics You Want

In Part 3, we looked at the basics of Petit Computer, including how to create graphics. Petit Computer has graphical capabilities that older versions of BASIC did not have, letting you position sprites freely, use the BG function to fill the screen with colorful backgrounds, and giving you tools to create characters. In this section, we will look at these functions in more detail.



※ The Sharp X1 was released in Japan in 1982 and had a character programming function or PCG.

4-01 Layering Characters & Backgrounds

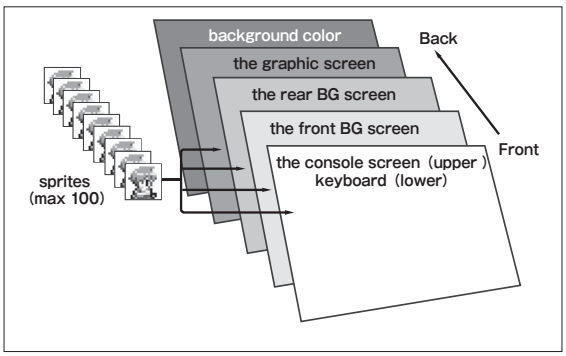
With Petit Computer, it's easy to create a scrolling background with characters moving freely on top of it. In this section, we'll take a look at how to program characters and backgrounds.

How the Display is Made Up

The screen display in Petit Computer is made up of a total of 5 different elements: the console screen/keyboard, the user BG screens, the sprites, the graphic screen, and the background color.

Each of these elements is suited to different uses, with sprites being ideal for game character display, and the BG screen being designed for the background display. The first layer of the display on the upper screen is the console screen, which the first layer on the lower screen is the keyboard.

It is important to understand how the different components of the display all fit together, particularly when learning how to program sprites and backgrounds.



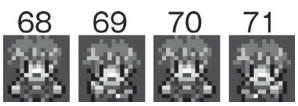
▲The display consists of the following five elements (from back to front): background color, the graphic screen, the rear BG screen, the front BG screen, and the console screen.

Character Display and Animation (Sprites)

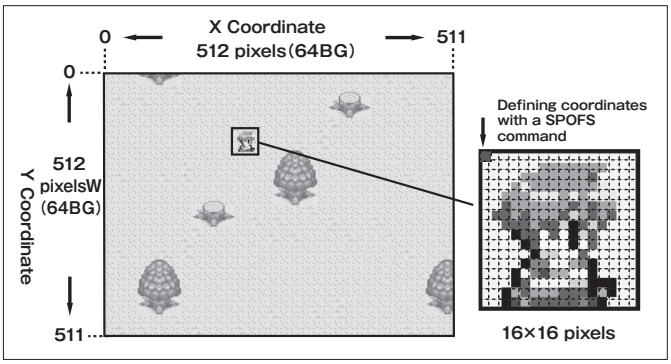
The sprite function lets you display characters on the screen, and allows you to move a large number of characters smoothly.

The SPOFS command is used to assign coordinates to sprites and display them. The upper and lower screens can each display up to 100 sprites. Each sprite can be up to 16x16 pixels in size, and can be one of 16 colors. You can select from a total of 16 palettes. For an overview of the palettes available, see page 10 of the Petit Computer Resources (Graphics) section.

The sprite function can also be used to display animated characters. The SPANIM command is used for animation, and allows you to assign character numbers which can be switched between in the on-screen display.



For example, by switching between the characters shown above, which have character numbers of 68→69→70→71, you can create an animated image of someone walking. Use the program below to make this character walk. The SPANIM command lets you set the character numbers to be switched between.

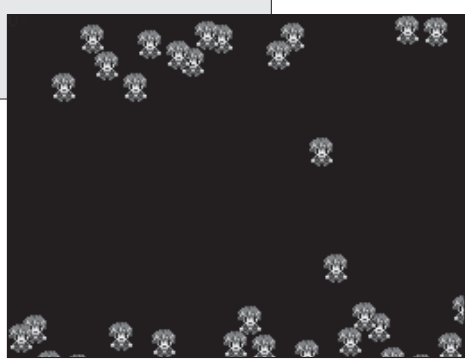


▲Each sprite can be up to 16x16 pixels in size. The coordinates for the sprite's location are set using the SPOFS command.

EXAMPLE 4-01: Animating a Sprite

```
0001 CLEAR
0002 DIM X(32),Y(32)
0003 CLS
0004 SPPAGE 0
0005 SPCLR
0006 FOR I=0 TO 31
0007 SPSET I,68,2,0,0,2          ←This is the initial sprite setting.
0008 SPANIM I,4,10               ←This is the animation display setting.
0009 X(I)=I*8                    ←This is the initial coordinate setting.
0010 Y(I)=96
0011 NEXT
0012 @MAIN
0013 FOR I=0 TO 31               ←The process is repeated from 0 ~31.
0014 IF RND(2) THEN Y(I)=Y(I)+1  ←The probability for sprite movement is set at 50%.
0015 IF Y(I)>191 THEN Y(I)=0
0016 SPOFS I,X(I),Y(I)           ←This causes the sprites to move.
0017 NEXT
0018 IF BUTTON()==0 THEN VSYNC 1
0019 GOTO @MAIN
```

When the program is run, 32 individual sprites will travel downwards to the bottom of the screen. Each character is animated so that it walks and moves its arms. Press SELECT to speed up the sprite movement.



▲Screen when Program is Run(EXAMPLE4-01)

Petit Computer includes a large number of sample characters which can be animated, so have fun experimenting with them all.

Processes Using Random Numbers

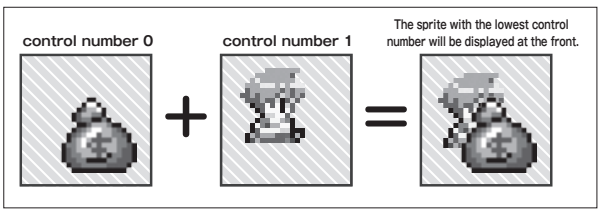
In the example program 4-01 shown above, the sprites have a 50% chance of moving. By using a RND function, you can generate a number at random. So for instance, IF RND(2) THEN～ will mean that a process has a fifty-fifty chance of occurring.

A Program with a 50% Probability

```
IF RND(2) THEN OOOOOO ←Enter the operation to take place if NO.
```

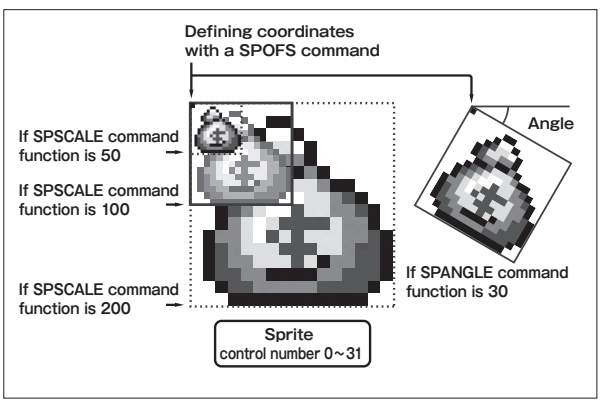
Sprite Scaling and Rotation

You can select the location where you wish sprites to be displayed. These locations include the front of the console screen, the front of BG screen 0, on the console screen, and more. When two or more sprites are displayed in the same location and their positions overlap, the one with the lowest assigned control number will be displayed on top. You can assign control numbers of 0-99 to sprites.



▲When sprites are displayed on the same screen and overlap, the sprite with the lowest control number will appear on top.

One of the special functions related to sprites is that using control number 0-31, you can scale or rotate sprites. If you want to shrink or enlarge a sprite, use the SPSCALE command, selecting a scaling ration of between 0-200%. If you want to rotate a sprite, use the SPANGLE command, which lets you turn the sprite between 0-359 °. The axis on which sprites are turned is at the top left of the sprite. See the image for an example of a sprite being rotated by 30°.



▲The axis around which sprites are rotated with the SPANGLE command, or scaled using the SPOFS command is set at the top left corner.

Use these functions to make fun programs which have plenty going on. The program below shows you how to move the axis around which a sprite rotates to the center of that sprite. It uses SIN and COS trigonometric functions to make it appear that the axis has moved to the center. Try using it in your own programs.

EXAMPLE 4-02: Rotating and Scaling Sprites

```
0001 GCLS                                ←Clears all graphics.
0002 GCIRCLE 128,96,20,15                ←Draws a circle.
0003 PNLTYPE "OFF"                       ←Deletes the keyboard on the lower screen.
0004 SPPAGE 0
0005 SPCLR
0006 X=128:Y=100:PAT=68:PAL=2            ←Sets coordinates and character number.
0007 SPSET 0,PAT,PAL,0,0,0               ←This is the initial sprite setting.
0008 SPANIM 0,4,30                       ←This is the animation display setting.
0009 @MAIN
0010 D=TCHX                              ←Enter angle of rotation.
0011 S=TCHY                              ←Enter rate of scaling.
0012 CLS
0013 PRINT "KAKUDO=";D
0014 PRINT "SCALE=";S
0015 A=RAD((D+225)% 360)
0016 R=8*S/100*SQR(2)
0017 X1=COS(A)*R                         ←Calculates amount of movement on x-axis at top left .
0018 Y1=SIN(A)*R                         ←Calculates amount of movement on y-axis at top left .
0019 SPANGLE 0,D                         ←This sets the angle of rotation.
0020 SPSCALE 0,S                         ←This sets the rate of scaling.
0021 SPOFS 0,X+X1,Y+Y1                   ←This calculates and displays the sprite movement in
                                           the top left.
0022 VSYNC 1                             ←Waits.
0023 GOTO @MAIN
```

When the program is run, a single sprite will be displayed in the center of the screen. Depending on the commands entered on the Touch Screen, it will get bigger or smaller, and rotate. Moving the stylus to the left or right will cause the sprite to rotate, while moving it up or down will shrink or enlarge it. The sprite's angle and scale are displayed in the top left of the screen. The sprite moves on its own, making it fun to watch.

▲Screen when Program is Run(EXAMPLE4-02)

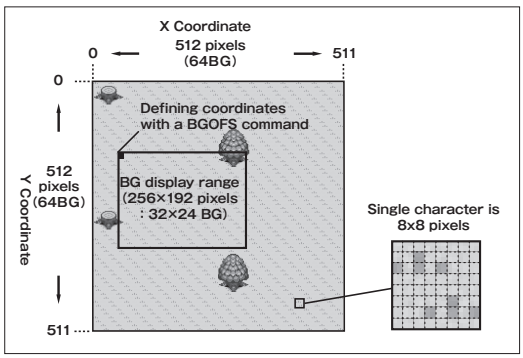
Creating Backgrounds (The BG Screen)

BG screen is short for 'background screen', and it refers to a function that is chiefly used for background display. The BG screen has two overlapping layers numbered 0 (the front layer) and 1 (the back layer). You can design and move each of these layers independently of the other. The same BG screen function can be used on both the upper and lower screens.

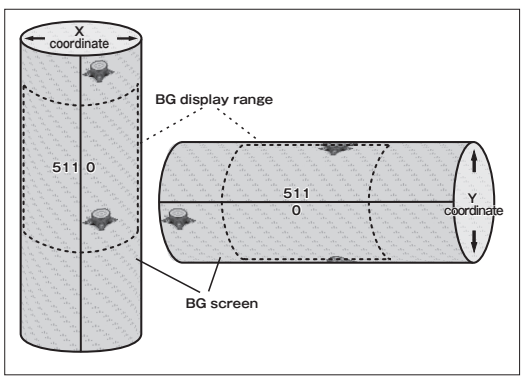
The BG screen is 512x512 pixels in size, and you can arrange characters on the screen in 8 pixel units. The display area is set at 256x192 pixels, though this can be modified using a BGCLIP command.

The BG screen is designed so that the top and bottom of the screen are connected, as are the left and right edges. This means that if you have a continuous image extending beyond the display area, you can make it scroll to appear that there is no limit to the background.

Here is an example of a program that causes the BG screen to scroll.



▲ Structure of BG Screen



▲ As both edges of the screen display are connected, the background will scroll on a continuous loop.

Scrolling on the BG Screen : program 4-03.

```
0001 BGPAGE 0
0002 BGCLIP 0,0,31,23          ←Sets size of display area.
0003 PAL=8                     ←Palette number.
0004 L=0                       ←Select BG0.
0005 FOR C=0 TO 1023
0006 BX=C%32                   ←Assign the remainder from C÷32.
0007 BY=FLOOR(C/32)            ←Assign integer value from C÷32.
0008 BGPUT L,BX,BY,C,PAL,0,0   ←Draw character.
0009 NEXT
0010 X=0:Y=0                   ←Coordinates of display area.
0011 @MAIN
0012 B=BUTTON()                ←Button input.
```

✚ To Next Page ✚

```
0013 IF B AND 1 THEN Y=Y-1          ←Loop coordinates.
0014 IF B AND 2 THEN Y=Y+1
0015 IF B AND 4 THEN X=X-1
0016 IF B AND 8 THEN X=X+1
0017 X=X AND 511:Y=Y AND 511        ←Set coordinates.
0018 BGOFS L,X,Y                    ←Display coordinates.
0019 CLS
0020 PRINT "X=";X;",Y=";Y
0021 VSYNC 1                        ←Wait.
0022 GOTO @MAIN
```

▲An image from example program 4-03.

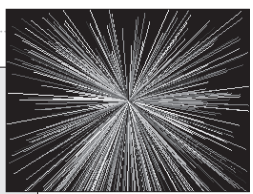
When the program is run, BG characters will be displayed on BG screen 0 on the upper screen. The X and Y coordinates assigned using the BGOFS command will be displayed on the top left of the screen. By pressing up, down, left or right on the +Control Pad, you can cause the BG screen to scroll in that direction. There is nothing programed to be displayed on BG screen 1 which is at the back, so it will remain black.

Layering Screens

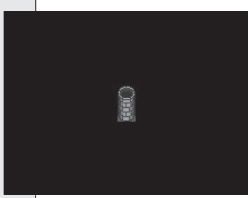
In this section, we will use what we have learned so far to create graphics by layering the console screen, graphic screen, BG screens, and sprites.

EXAMPLE 4-04: BG Screen Scrolling

```
0001 VISIBLE 1,1,1,1,1,1
0002 '-------------------GRAPH     ←Display on graphic screen.
0003 GCLS
0004 FOR I=1 TO 500
0005 X=RND(256):Y=RND(256)
0006 GLINE 128,96,X,Y,RND(256)
0007 NEXT
0008 '-------------------BG0       ←Display on BG screen 0.
0009 PAL=8
0010 L=0:X=15:C=512+48
0011 FOR Y=10 TO 13                 ←Display chimney character.
0012 BGPUT L,X  ,Y,C  ,PAL,0,0
0013 BGPUT L,X+1,Y,C+1,PAL,0,0
0014 C=C+32
```



▲Graphics (from back of screen)



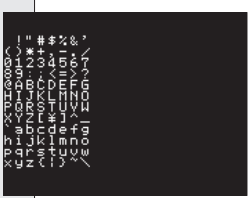
▲BG screen 0 (front)

✚ To Next Page ✚

```
0015 NEXT
0016 '--------------------BG1          ←Draw image on BG1
0017 L=1
0018 FOR Y=10 TO 21
0019 FOR X=4  TO 27                   ←Display grass on BG screen
0020 C=32+RND(2)
0021 BGPUT L,X,Y,C,PAL,0,0
0022 NEXT
0023 NEXT
0024 '--------------------SPRITE       ←Sprite settings.
0025 SPCLR
0026 X=120:Y=64:PAL=2
0027 SPSET 0,68,PAL,0,0,2             ←Hero character settings.
0028 SPOFS 0,X,Y
0029 SPSCALE 0,200
0030 '--------------------CONSOLE      ←Display on console screen.
0031 CLS
0032 FOR I=&H20 TO &H7F               ←Display characters as test.
0033 LOCATE I%8,FLOOR(I/8)
0034 PRINT CHR$(I)
0035 NEXT
0036 '--------------------CHANGE MODE  ←Wait for button input.
0037 M=0
0038 @MAIN
0039 IF BUTTON()==0 THEN @MAIN
0040 M=(M+1)%6                        ←Loop within range 0-5.
0041 IF M==0 THEN VISIBLE 1,1,1,1,1,1 ←Switch between
0042 IF M==1 THEN VISIBLE 1,0,0,0,0,0   screen display.
0043 IF M==2 THEN VISIBLE 0,0,1,0,0,0
0044 IF M==3 THEN VISIBLE 0,0,0,1,0,0
0045 IF M==4 THEN VISIBLE 0,0,0,0,1,0
0046 IF M==5 THEN VISIBLE 0,0,0,0,0,1
0047 VSYNC 30                         ←Wait.
0048 GOTO @MAIN
```

▲BG screen 1 (rear)

▲Sprite



▲Console (screen front)

The screenshot on the right shows the program when it is running. As you can see, a complex array of images has been built up by layering the different display elements. This process allows you to create an image such as this one, where the hero character is passing behind an obstacle.

By pressing any button, with the exception of SELECT, you can switch the display as follows: console screen → BG screen 0 → BG screen 1 → Sprite → graphic screen → all screens. It's fun to see which images are displayed on each screen.

▲Put them all together and...

4-02 Getting to Grips with the Tools

There are three graphic tools included in Petit Computer: a character editor, a BG screen editor, and a graphic editor. These are all large-scale programs created using BASIC. You can make use of these tools to create your own characters and backgrounds and display them in your own programs.

Getting to Grips with the Resources

Before looking at each of these tools in detail, let's look at the resources available to you.

The kind of files	The name of Resources	Contents
PRG	PRG	Program
MEM	MEM	Memory
COL	COL0～COL2	
GRP	GRP0～GRP1	Graphics (0=Upper Screen, 1=Lower Screen)
SCR	SCU0～SCU1	User BG screens (0=front, 1=rear)
CHR	BGU0～BGU3	User BG characters (bank 0～3)
	SPU0～SPU7	User sprite character (bank 0～7)

In addition to loading programs, Petit Computer also allows you to load files containing characters and much more. By using the LOAD and SAVE commands along with "Resource Name: File Name", you can gain access to a wealth of resources. See the table above for an overview of the types of resource available.

You can use the three editing tools we will look at below to create graphic and BG screens, as well as character files. All of these can be saved so that they can be later loaded and used as resources in your own programs.

Saving and Loading MEM Resources

There are other resources besides program and graphic resources which can be saved and loaded in file form. In this section, we will look at one of these resources, a memory (MEM) program that allows data to be saved and loaded. In the example below, the number of times you run a program will be counted.

EXAMPLE4-05 : Counter will be stored even after you switch the power off.

```
0001 LOAD "MEM:MEMCNT"          ←Loads file. The contents will be stored to the variable MEM$.
0002 C=VAL(MEM$)+1              ←The contents will be converted into numerical form, 1 will be added and assigned to variable C.
0003 PRINT "ｼﾞｯｺｳｶｲｽｳ ";C      ←Displays total number of times program has been run.
0004 MEM$=STR$(C)               ←Assigns a character string to variable MEM$.
0005 SAVE "MEM:MEMCNT"          ←Saves MEM$ as a file.
```

```
OK
LOAD"J405
OK
ｼﾞｯｺｳｶｲｽｳ 1
OK
ｼﾞｯｺｳｶｲｽｳ 2
OK
ｼﾞｯｺｳｶｲｽｳ 3
OK
ｼﾞｯｺｳｶｲｽｳ 4
OK
ｼﾞｯｺｳｶｲｽｳ 5
OK
ｼﾞｯｺｳｶｲｽｳ 6
OK
ｼﾞｯｺｳｶｲｽｳ 7
```

When this program is run, the total number of times it has been run will be displayed on the screen. This total will be saved as a file, meaning that it will be stored even after you switch the power off. By making use of this function, you can keep records of high scores on games you create. A confirmation dialog box is displayed each time a LOAD or SAVE command is used, and this may get tiresome after a while. You can enter LOAD "MEM:MEMCNT",0 to reduce the number of dialogue boxes displayed by one.

CHARACTER EDITOR

Character Editor lets you create characters to be used on BG screens or as sprites.

There is a huge variety of character data pre-loaded onto Petit Computer, but if you want to create your own original games, it's likely that you'll want the freedom to design your own characters, and that's where this tool comes in.

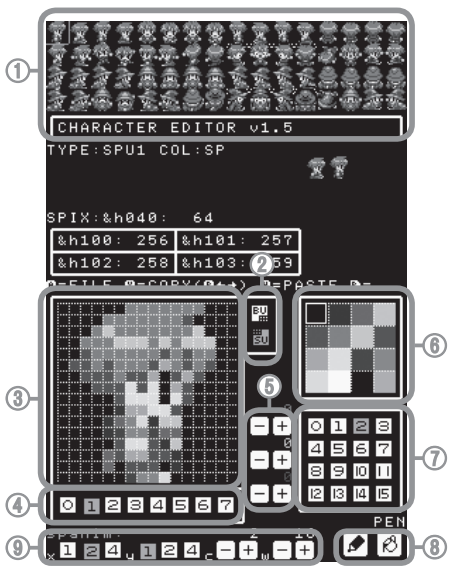
When Petit Computer was first released, there was a bug which did not allow users to save resources. However this bug was fixed on the Petit Computer ver.1.1 update released in Japan on 16th June 2011 which included the latest character editor.

How to Use This Tool

The file name for this tool is CHRED. Enter the following command in Run Mode to run the tool.

```
EXEC"CHRED"↵
```

The screen display



① : The entire character set in a single bank will be displayed. The flashing frame indicates your currently selected character.

② : Select the type of character you want to work on. Touch the screen to switch between the different types.
「BU」…BG screen characters
「BG」…Sprite characters

③ : The edit area is 16x16 pixels. Use the Touch Screen to draw in it.

④ : The currently selected bank of characters. BG screen banks will display 0-3, while sprite banks will display 0-7.

⑤ : RGB value of the selected color. Touch to change.

⑥ : Color samples with color codes of 0-15. Touch to select.

⑦ : Color palettes 0-15. Touch to select.

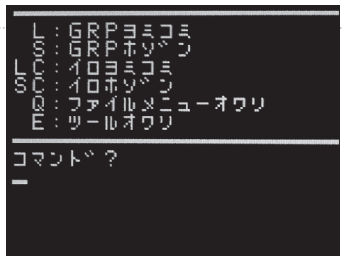
⑧ : Tool selection. Touch to change.
「PEN」…Draw with stylus.
「PAINT」…Fill.

⑨ : If you have used this command to animate a character, you can see the order of the character display here. Touch x and y to set the number of times a character is displayed, touch c to set the number of characters, and touch w to adjust display timing.

Button Controls

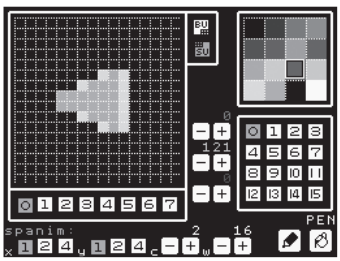
A Button : Go to file menu
+Control Pad: Move edit area
Y Button : Delete character you're working on in edit area

File Menu



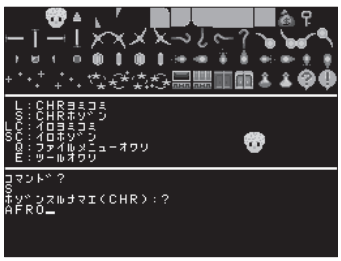
「**L** ⏎」: Load CHR file and display as bank you're currently editing
「**S** ⏎」: Save character you're drawing along with its bank as CHR file
「**LC** ⏎」: Load COL file and set as color palette
「**SC** ⏎」: Save color palette as COL file
「**Q** ⏎」: Close menu and go back
「**E** ⏎」: End program

Creating a Character



▲Refer to ② in the previous diagram and choose whether to work on a BG or sprite character. Select the color palette you wish to work on by referring to ⑦.

▲Select the character you wish to work on from ① and then start drawing.



▲Press the A Button to access the file menu and use the S ⏎ command to enter a file name and save. If you have created your own colors with ⑤, use an SC ⏎ command to save your palette.

Saving Characters

By entering an S ⏎ command or an MCHR ⏎ command, character data will be saved in this format: "CHR:MYCHR".

If you wish to load this as a BG screen character on the upper screen (bank 0), run the following command:

```
LOAD "BGU0:MYCHR" ⏎
```

If you wish to load it as a sprite character on the upper screen (bank 0), run the following command:

```
LOAD "SPU0:MYCHR" ⏎
```

※If you are having trouble loading data, run SPPAGE 0 prior to using the LOAD command.

When you run the LOAD command, the character data in the file will be set. You can then use CHRED to confirm how the loaded character data will be displayed.

It can be fun to play around with existing game character data and see how you can adapt it.

SCREEN EDITOR

This tool lets you place characters on the BG screen and create backgrounds for your games and other programs. Having the right background can make a big difference in the feel of role-playing games or shooting games. This tool is sure to prove invaluable when you are creating your own games.

How to Use This Tool

The file name for this tool is CHRED. Enter the following command in Run Mode to run the tool.

```
EXEC "SCRED"↵
```

When you run this tool, the following question is displayed:

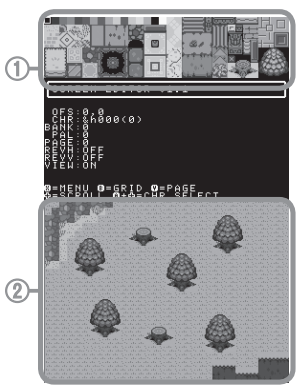
```
ｷｬﾗﾃﾞｰﾀﾆｺﾐﾏｽｶ(Y/N)?
```

To use character data you have created yourself, select Y ↵ and then enter the data file name. If you don't wish to use your own data, select N ↵ and get started with the editing tool.

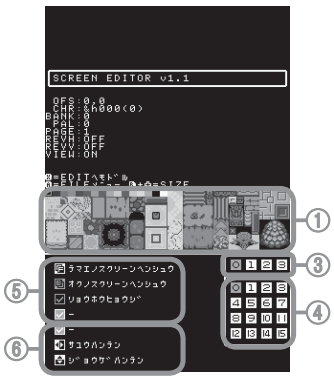
Display Structure

①: This is a sample of a BG character. A bank of characters are displayed in the edit screen menu.
②: The edit area. Touch this to copy it to the BG screen.
③: BG character banks. Touch to select.
④: Color palettes 0-15. Touch to select.
⑤: Switch between layers of BG screen display
⑥: Rotate vertically and horizontally. The orange color indicates that this function is being used.

Edit Screen



Menu Screen



Button Controls

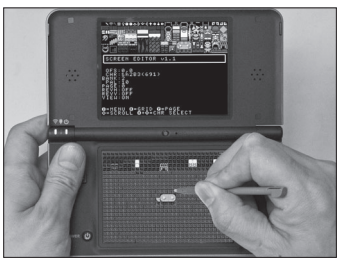
X Button : Switch to menu and edit screen
B Button(Edit Screen) : Display grid
+Control Pad(Edit Screen) : Move edit area
A Button(Menu Screen) : Go to file menu
R Button&+Control Pad : Modify copy area
Y Button : Select front or rear BG screen

File Menu

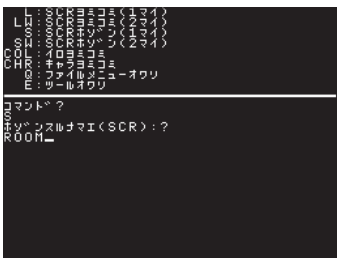
「**L** ⏎」: Load an SCR file and display it on the upper BG screen, as the layer being edited
「**LW** ⏎」: Load two SCR files and display on the upper BG screen, as BG screens 0 and 1
「**S** ⏎」: Save the BG screen currently being edited on the upper screen as an SCR file
「**SW** ⏎」: Save both BG screens 0 and 1 on the upper screen as SCR files
「**COL** ⏎」: Load a COL file and set as color palette on BG screen
「**CHR** ⏎」: Load a CHR file an set as character on BG screen
「**Q** ⏎」: Close menu and go back
「**E** ⏎」: End program

How to Create a BG Screen

▲Select whether to edit the front or back BG screen and select the characters you wish to use.



▲Press the X Button to go to the edit screen. Use the edit area to draw or modify characters.



▲On the menu screen, press the A Button to go the file menu. Use the S ⏎ command to enter the file name and save. If you have edited both the front and rear BG screens, you can use the SW ⏎ command to save both screens.

● Tips on Drawing

The background is made up of two separate BG screen layers, which are laid over each other. By placing a character on the front layer, and then place a character behind it on the rear layer. This can be made to look like the rear character is hiding behind the one at the front.

You can use each layer of the background in different ways. For instance, you could place tree trunks or leaves on the front screen, and grass on the rear screen. If you then place sprites in between the two screens, it can look like the player character is hiding amongst the undergrowth.

Saving BG Screen Data

When you enter S ⏎ or MYSC ⏎ commands on the file menu, the screen data will be saved to an "SCR:MYSC" file.

If you wish to load this to BG screen 0 on the upper screen, run the following command:

```
BGPAGE 0
LOAD"SCU0:MYSC"⏎
```

When the LOAD command has run successfully, the contents of the file will be displayed on the upper screen. If nothing is displayed on the screen, it may be because the BG screen is not allowed to be displayed. In this case, try running the following command:

```
VISIBLE 1,1,1,1,1,1⏎
```

GRAPHIC EDITOR

There are limits to how much can be achieved by using program commands to draw images on the graphic screen. By limiting yourself to setting coordinates and drawing lines and curves, it can be very tough to come up with a program with impressive visuals. This is where the Graphic Editor comes in. You can make full use of its drawing tools, or hand-draw images and then save the results in a file.

How to Use This Tool

To load the tool, run the following command:

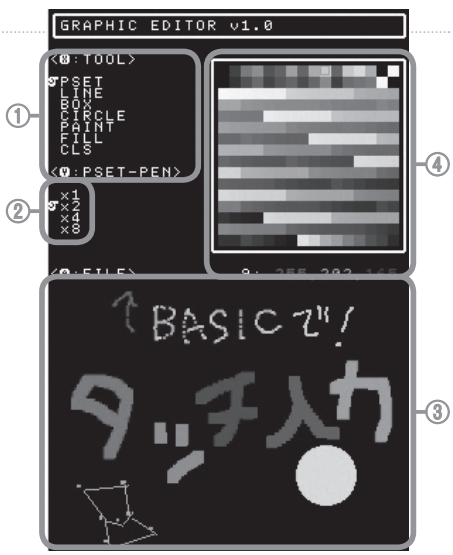
```
EXEC "GRPED"↵
```

After running this command, or when you leave the file menu, the following question will be displayed:

```
ｶﾞﾒﾝｦｹｼﾏｽｶ(A=YES)
```

Press the A Button to clear the screen. If you press any other button, you will continue with the lower screen display unchanged. When you wish to start with a blank canvas, be sure to press the A Button.

The screen display



① : Tool. Press the X Button to select
Draw a point　Draw a line　Draw a square
Draw a circle　Paint area one color　Fill rectangle
Erase everything on screen
② : Pen size. Select with Y Button
③ : Edit area. Touch the screen or slide stylus to draw.
④ : Color samples. Select with +Control Pad. The flashing frame indicates the selected color. The color code (0-255) and R, G, and B value is displayed below.

Button Controls

X Button : Switch tools
Y Button : Switch pen size
+Control Pad : Choose from 256 colors
R Button : Grid display ON/OFF
A Button : Switch to file menu

File Menu

When you press the A Button to open the file menu, you will be asked whether or not you wish to copy your image to the upper screen.

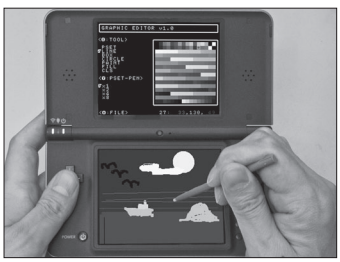
Enter Y ↵ to copy the graphic image on the lower screen to the top screen.

Enter N ↵ to continue to the file menu without copying the image.

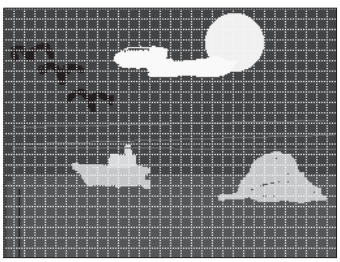
「**L**↵」: Loads GFP files to the graphic screen on the lower screen.
「**S**↵」: Saves the graphics on the lower screen to a GRP file.
「**LC**↵」: Loads COL file and sets it as the color palette when creating graphic images.
「**SC**↵」: Save graphic color palette to a COL file.
「**Q**↵」: Exit menu and go back.
「**E**↵」: End program.

*After leaving the file menu, you will be asked the same question that appeared when you accessed it.

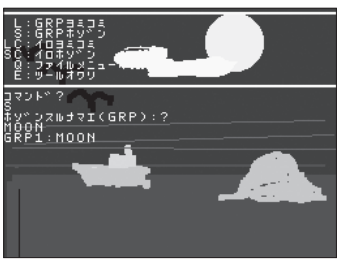
How to Create a Graphic Screen



▲Press the X Button to select tools from ①, and the Y Button to select thickness from ②. Use the +Control Pad to select from ④ and start to draw.



▲Press the R Button to display the grid, giving you more precision when drawing.



▲Press the A Button to access the file menu and use the S ↵ command to enter a file name and save.

Saving Graphic Data

Entering S ↵ or MYGRP ↵ on the file menu will save your graphic data in a "GRP:MYGRP" format.

If you wish to display this graphic data on the upper screen, run the following command:

```
LOAD"GRP0:MYGRP"↵
```

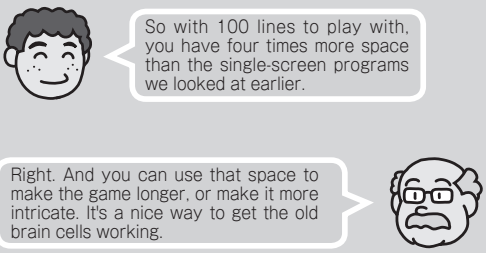
When the LOAD command has run successfully, the contents of the file will be displayed on the upper screen. If you wish to display it on the lower screen, change GRP0 to GRP1 in the command above.

Special Section

What Can You Create in a Limited Space?

100-Line Programming Corner

Following on from the Single-Screen Programming Corner that began on page 64, we're now going to take up the gauntlet of programming within a limitation of 100 lines. Even within these limitations, you can come up with programs that have the flavor and feel of the classic BASIC programs of the 1980s. In the following section, we will introduce three programs which the author of this guide came up with.



Entry No.1

Race a Space Craft and Aim for the Fastest Time

G-ZERO

●Introducing the Program

This game is set in the zero gravity conditions of space. The direction your space craft will fly is governed by inertia. One nice feature of this game is that the screen will scroll 360 ° in every direction.

●How it Works

You will fly your craft round the zero-gravity track. Pressing left and right on the +Control Pad will alter your direction, while pressing the A Button will cause you to accelerate. Your lap time is displayed each time you complete a circuit of the track, and your aim is to set an unbeatable personal best.

Program List

```
0001 '---------- G-ZERO
0002 CLS
0003 CLEAR:DIM M$(8)
0004 M$(0)="◤       ◥■"
0005 M$(1)="  ◢■■■◣  ■"
0006 M$(2)="  ■■■■■  ■"
0007 M$(3)="  ■■■■■  ■"
0008 M$(4)="  ■■■■◤  ■"
0009 M$(5)="  ◥■◤   ◢■"
0010 M$(6)="◣     ◢■■■"
0011 M$(7)="■■■■■■■■"
0012 '---------BG0
0013 BGPAGE 0
0014 L=0
0015 FOR Y=0 TO 7
0016 FOR X=0 TO 7
0017 C=ASC(MID$(M$(Y),X,1))
0018 CHRREAD("BGF0",C),BF$
0019 FOR J=0 TO 7
0020 FOR I=0 TO 7
0021 A=VAL("&H"+MID$(BF$,I+J*8,1))
0022 BX=I+X*8:BY=J+Y*8
0023 C=RND(4)+363:PAL=11
0024 IF A THEN C=RND(4)+378:PAL=3
0025 BGPUT L,BX,BY,C,PAL,0,0
0026 NEXT
0027 NEXT
0028 NEXT
0029 NEXT
0030 C=12
0031 FOR BX=24 TO 31
0032 FOR BY=0 TO 7
0033 BGPUT L,BX,BY,C,PAL,0,0
0034 NEXT
0035 NEXT
0036 '--------PLAYER INIT
0037 PAL=2
0038 SPPAGE 0:SPCLR
0039 SPSET 0,176,PAL,0,0,1
0040 SPSET 1,251,PAL,0,0,1
0041 R=8*SQR(2)
0042 Z=0:F=1:T1=0
0043 X=128:Y=32:X1=0:Y1=0
```

Track data

←Read font data.

←Draw background.

←Fill the section of track where the finish line is located.

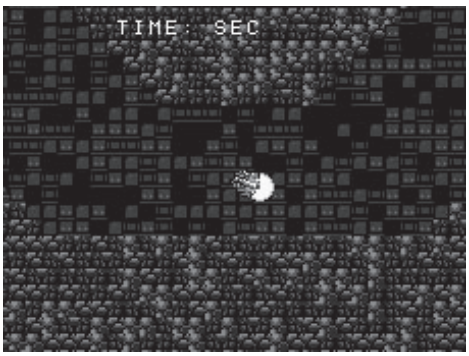
Initial sprite setting

✚ To Next Page ✚

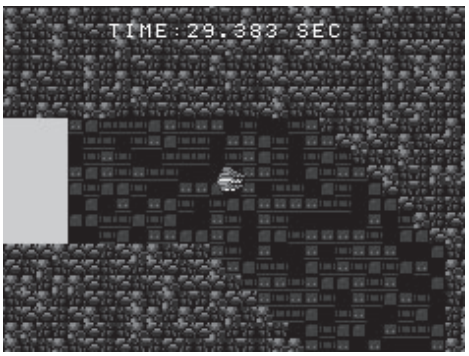
```
0044 A=180:A1=2:SMAX=8                    ←Variable A1 is degree of rotation. Variable
0045 '--------MAIN                          SMAX is the maximum speed.
0046 @MAIN
0047 B=BUTTON()
0048 K=0.99                               ←Ratio used to maintain speed of craft.
0049 IF B AND 8 THEN A=A+A1
0050 IF B AND 4 THEN A=A-A1
0051 IF B AND 15 THEN K=0.98
0052 IF A<0    THEN A=A+360
0053 IF A>=360 THEN A=A-360
0054 X1=X1*K:Y1=Y1*K                      ←Decelerate.
0055 S=(X1*X1)+(Y1*Y1)
0056 MX=-8:MY=-8
0057 IF (B AND 16)==0 THEN @MOV0
0058 IF S>SMAX THEN @MOV0
0059 T=RAD((A+180)% 360)
0060 X1=X1+COS(T)/20                      ├ Accelerate.
0061 Y1=Y1+SIN(T)/20
0062 MX=COS(RAD(A))*Z+128
0063 MY=SIN(RAD(A))*Z+96
0064 Z=Z+1:S=S*700
0065 IF Z>16 THEN Z=6:BEEP 1,S            ←Play sound effect when craft accelerates.
0066 @MOV0
0067 SPOFS 1,MX-8,MY-8                    ←Display fire when craft accelerates.
0068 SPANGLE 0,A
0069 T=RAD((A+225)% 360)
0070 MX=COS(T)*R+128
0071 MY=SIN(T)*R+96
0072 SPOFS 0,MX,MY
0073 X=X+X1:Y=Y+Y1
0074 BGREAD(L,X/8,Y/8),C,PAL,H,V
0075 IF C>=378 THEN GOSUB @BOUND          ←Process when craft collides with wall.
0076 BX=(X+(512-128))%512                 ├ Obtain x and y coordinates for BGOFS.
0077 BY=(Y+(512-96))%512
0078 BGOFS L,BX,BY
0079 VSYNC 1
0080 GOSUB @TCHK                          ←Process that measures race time.
0081 GOTO @MAIN
0082 '----------
0083 @BOUND
0084 X1=-X1:Y1=-Y1:BEEP 13
0085 RETURN
0086 '----------TIME
0087 @TCHK
0088 I=FLOOR(X/64):J=FLOOR(Y/64)
```

✚ To Next Page ✚

```
0088 IF I!=3 THEN RETURN                          ←Inverts rate of acceleration.
0089 IF J!=6 AND J!=0 THEN RETURN
0091 IF J==F THEN RETURN
0092 F=J
0093 IF F!=0 THEN RETURN
0094 BEEP 7
0095 T2=MAINCNTL                                  ←System variable used to obtain frame rate.
0096 LOCATE 8,2:PRINT "TIME:";
0097 IF T1 THEN PRINT (T2-T1)/60;                 ←Frame number is divided by 60 and
0098 PRINT " SEC    ":T1=T2                         converted into number of seconds.
0099 RETURN
```



▲Control your space craft and complete laps of the course. Once you get the hang of it, you'll even be able to use drifting techniques.



▲When you reach the finish line, your time will be displayed. You're competing with yourself, so aim to set an unbeatable personal best.

Check Point

Hey! Are you telling me there are no rival space crafts? Do you just fly round by yourself forever?

Well, it is a 100-line program after all! Why not start by adjusting the layout of the track, and the parameters for acceleration and turning?

I'll teach you a good trick. You can fly the wrong way round the track, and it will still record your lap time. Seems the judges aren't too harsh here.

Entry No.2

Meet the Woman of Your Dreams

BASIC Romance

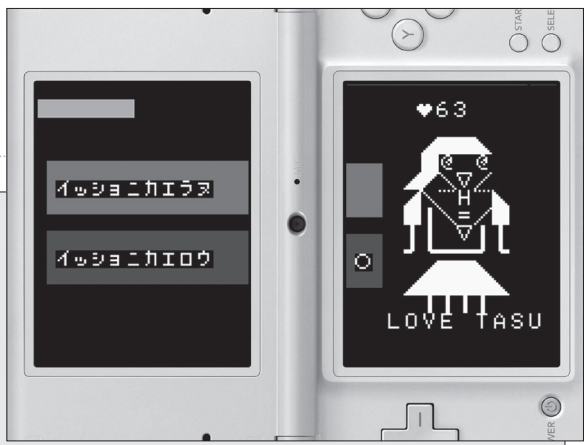
●Introducing the Program

This program is not to be taken too seriously. You play this game by turning your DSi or 3DS system on its side and talking to your beautiful virtual girlfriend.

●How it Works

Rotate your DSi or 3DS system 90° so it looks like you have a photo of your beloved girlfriend. The rules are very simple. You select one of two different responses to your girlfriend by touching the screen, and the goal is to make her happy. You will face five questions. The more rapidly you respond, the more hearts you will rack up.

▶ Give things an innovative twist by rotating the screen 90°.



Program List

```
0001 '----------LOVE TASU
0002 CLS:PNLTYPE "OFF"
0003 @START
0004 GOSUB @GC:GCLS 14
0005 X=208:Y=36:BGMSTOP
0006 RESTORE @GIRL
0007 FOR K=1 TO 11
0008 READ A$:GOSUB @TXT1
0009 NEXT
0010 @GIRL
0011 DATA "   ◢■■◣  "
0012 DATA "   ■@ @  "
0013 DATA "  ◢◤＼▽／ "
0014 DATA "  ◢＼￣H￣◣"
0015 DATA "  ■❙＼=／■"
0016 DATA "  ❙❙ ▽❙❙"
0017 DATA "  ┘└──┘└"
0018 DATA "   ◢■■◣  "
0019 DATA "  ◢■■■■◣"
0020 DATA "   ❙❙❙❙  "
0021 DATA "LOVE TASU"
0022 LV=0:CNT=5:MAX=8
```

Girlfriend display data.

←Total question number assigned to the variable MAX. Variable CNT shows the number of questions asked.

✚ To Next Page ✚

```
0023 GOSUB @TOUCH
0024 '--------------------
0025 @MAIN
0026 GOSUB @GC:BGMPLAY 2:VSYNC 90
0027 RESTORE @MESSAGE
0028 N=RND(MAX):J=RND(2)
0029 FOR I=0 TO N
0030 READ A$,B$:NEXT
0031 IF J THEN T$=A$:A$=B$:B$=T$      ←Shuffle the correct responses.
0032 M$(0)=A$:M$(1)=B$
0033 FOR K=0 TO 1
0034 X=136-(K*64):Y=8:C=K*2+2
0035 GPAGE 0:GFILL X,Y,X+48,192,C
0036 GPAGE 1:GFILL X,0,X+48, 32,C
0037 X=X+30:Y=Y+8
0038 A$=M$(K):GOSUB @TXT0
0039 NEXT
0040 BGMPLAY 24:W=192:X=224:Y=0
0041 GFILL X,Y,X+16,Y+W,13
0042 @KWAIT
0043 VSYNC 2:BEEP 0:W=W-4
0044 GFILL X,Y+W,X+16,Y+192,0
0045 IF TCHST==0 THEN @KWAIT
0046 A$="○" :I=1-FLOOR(TCHX/128)
0047 IF I==J THEN @SEIKAI             ←Evaluate response.
0048 A$="×" :BEEP 13:W=0
0049 @SEIKAI
0050 X=166-(64*I):Y=8:GOSUB @TXT1
0051 FOR K=1 TO FLOOR(W/4)
0052 X=240:Y=64:LV=LV+1
0053 A$="♥"+STR$(LV):GOSUB @TXT1
0054 BEEP 67:VSYNC 2
0055 NEXT
0056 VSYNC 60
0057 CNT=CNT-1:IF CNT THEN @MAIN
0058 BGMPLAY 11:VSYNC 180             ┐
0059 GOSUB @TOUCH                     ├ Process for ending the game.
0060 GOTO @START                      ┘
0061 '--------------------
0062 @TXT1                            ┐
0063 GPAGE 1:GOTO @TXT                │
0064 @TXT0                            │
0065 GPAGE 0                          │
0066 @TXT                             │
0067 FOR I=0 TO LEN(A$)-1             │
```

+ To Next Page +

```
0068 C=ASC(MID$(A$,I,1)):P=0
0069 CHRREAD("BGF0",C),BF$
0070 FOR TX=X TO X-15 STEP -2
0071 FOR TY=Y TO Y+15 STEP 2
0072 V=VAL("&H"+MID$(BF$,P,1))
0073 P=P+1:IF V==0 THEN V=14
0074 GFILL TX,TY,TX-1,TY+1,V
0075 NEXT
0076 NEXT
0077 Y=Y+16
0078 NEXT
0079 X=X-16:Y=Y-LEN(A$)*16
0080 RETURN
0081 '------------------
0082 @TOUCH
0083 GOSUB @GC:X=16:Y=0
0084 A$="TOUCH PANEL":GOSUB @TXT0
0085 @SWAIT
0086 IF TCHST==0 THEN @SWAIT
0087 @GC
0088 GPAGE 0:GCLS:BEEP 11
0089 GPAGE 1:GFILL 0,0,255,36,14
0090 RETURN
0091 '---------------MESSAGE DATA
0092 @MESSAGE
0093 DATA "ｲｯｼｮﾆｶｴﾛｳ","ｲﾝﾄﾞﾆｶｴﾛｳ"
0094 DATA "ｲｯｼｮﾆｶｴﾛｳ","ｲｯｼｮﾆｶｴﾗﾇ"
0095 DATA "ｲｯｼｮﾆｶｴﾛｳ","ｲｯｼｮﾆｴﾛｶｳ"
0096 DATA "ｷﾚｲﾀﾞﾈ","ｷﾗｲﾀﾞﾈ"
0097 DATA "ｴｶﾞｵｶﾞｽﾃｷ","ｴｶﾞｵｶﾞﾌﾞｷﾐ"
0098 DATA "ｷﾐｶﾞｽｷ","ｱﾝｷﾓｶﾞｽｷ"
0099 DATA "ｷﾐｼｶｲﾅｲ","ｷﾐﾊｲﾗﾅｲ"
0100 DATA "ﾒｰﾙｵｸﾙﾖ","ﾒｰﾙﾒﾝﾄﾞｲ"
```

Subroutine that rotates character display by 90°

Question data. Repeats order for correct and incorrect responses.

Check Point

But the graphics are so dated! I'm afraid she's just not my type.

I do wish they'd made her a bit cuter.

Yes, I'm afraid I can't see a relationship budding with this young lady. There aren't many questions either. I'd say there's plenty of room for improvement in this program.

Entry No.3

This Game's a Blast!

BASIC Bomber

●Introducing the Program

This is an old-fashioned game that utilizes parabolic calculations. You will set the angle and speed using the Touch Screen and then throw bombs at the target.

●How it Works

This is a 2D game in which gravity plays a big part. You determine the angle and speed of your bomb using the Touch Screen, before throwing it and trying to take out enemies with the blast. The game ends when you use all 10 of the bombs you are equipped with. The bomb will not damage the enemies by just hitting them, so you need to get the timing of the blast just right.

Program List

```
0001 '----------------ﾊﾞｸﾀﾞﾝﾅｶﾞ
0002 CLS:PNLTYPE "OFF"
0003 @RETRY
0004 SPPAGE 0:SPCLR
0005 SPSET 0,142,4,0,0,0
0006 SPANIM 0,2,20:SPOFS 0,64,99
0007 SPSET 1,196,6,0,0,0
0008 SPANIM 1,2,20:SPOFS 1,164,99
0009 GOSUB @TITLE
0010 @START
0011 GOSUB @MAPINIT
0012 AX=8:AY=184:TX=AX:TY=AY
0013 CNT=10:GOSUB @HOUKOU          ←The number of bombs you have it assigned
0014 Y=200:B=0                        to variable CNT.
0015 EX=RND(140)+80:EY=RND(99)+64  ←Random values are used to assign
0016 SPOFS 1,EX-8,EY-8:SPCHR 1,196    coordinates where enemies will be
0017 '-------------MAIN                displayed.
0018 @MAIN
0019 VSYNC 1:SPOFS 0,X-8,Y-8
0020 IF B==0 THEN @FIRE
0021 C=2:B=B-1:IF B THEN @BOUND
0022 GCIRCLE X,Y,30,C:GPAINT X,Y,C ←Graphic display of bomb blast.
0023 Y=200:BEEP 13
0024 IF GSPOIT(EX,EY)==C THEN @CL
0025 VSYNC 40:GCLS 246:GOTO @MAIN
0026 @BOUND
```

✚ To Next Page ✚

```
0027 GOSUB @BGCHKX
0028 IF C THEN X1=-X1:BEEP 8
0029 GOSUB @BGCHKY
0030 IF C THEN Y1=-Y1:BEEP 8
0031 IF Y1<7 THEN Y1=Y1+0.1
0032 X=X+X1:Y=Y+Y1
0033 IF Y>200 THEN B=0
0034 GOTO @MAIN
0035 @FIRE
0036 IF CNT==0 THEN @GOVR
0037 IF TCHST==0 THEN @MAIN
0038 CNT=CNT-1:TX=TCHX:TY=TCHY
0039 GOSUB @HOUKOU
0040 X=AX:Y=AY:T=ATAN(TY-Y,TX-X)
0041 CX=COS(T):CY=SIN(T)
0042 TX=TX-X:TY=TY-Y
0043 R=SQR((TX*TX)+(TY*TY))/20
0044 IF R>7 THEN R=7
0045 X1=CX*R:Y1=CY*R
0046 B=190:BEEP 14:GOTO @MAIN
0047 @CL       '-----STAGE CLEAR
0048 SPCHR 1,252:BGMPLAY 9
0049 VSYNC 180:LV=LV+1:GOTO @START
0050 @GOVR     '-----GAME OVER
0051 BGMPLAY 5:A$="GAME OVER"
0052 GOSUB @TWAIT:GOTO @RETRY
0053 @HOUKOU   '-----HOUKOU
0054 GPAGE 1:GCLS 14
0055 GCIRCLE AX,AY,140,4
0056 GLINE AX,AY,TX,TY,2
0057 X=160:Y=72:A$=STR$(CNT)+"ﾊﾝ"
0058 GOSUB @PUTSTR:GPAGE 0
0059 RETURN
0060 @BGCHKY   '-----BG CHECK
0061 CX=X:CY=Y-7+Y1:CX1=0:CY1=14
0062 GOTO @BGCHK
0063 @BGCHKX
0064 CX=X-7+X1:CY=Y:CX1=14:CY1=0
0065 @BGCHK
0066 C=0:FOR I=1 TO 2
0067 BX=FLOOR(CX/8) AND 511
0068 BY=FLOOR(CY/8) AND 511
0069 BGREAD(L,BX,BY),BC,PAL,H,V
0070 CX=CX+CX1:CY=CY+CY1
0071 C=C OR BC:NEXT:RETURN
```

Deflection when bomb hits an obstacle.

←Increase of gravitational acceleration.

←ATAN function used to obtain angle of line.

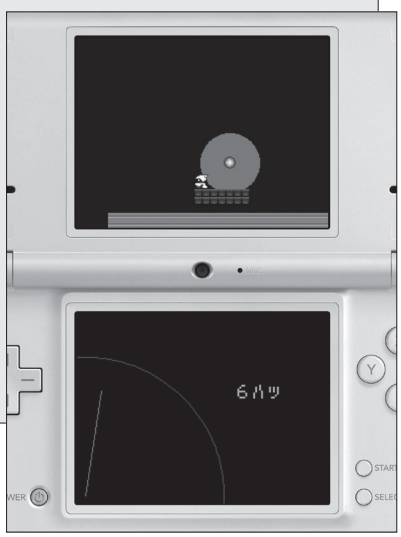
←Obtains values for X and Y distance.

←Collision detection for bomb and background.

✚ To Next Page ✚

```
0072 @MAPINIT   '------INIT MAP
0073 GPAGE 0:GCLS 246:BGMPLAY 7
0074 X=RND(20)+10:Y=RND(11)+10
0075 C=57:W=4:H=2:GOSUB @BGFILL
0076 X=4:Y=22
0077 C=92:W=28:H=2:GOSUB @BGFILL
0078 RETURN
0079 @BGFILL    '------BG FILL
0080 FOR TX=0 TO W-1
0081 FOR TY=0 TO H-1
0082 BGPUT L,X+TX,Y+TY,C,PAL,0,0
0083 NEXT:NEXT:RETURN
0084 @PUTSTR    '------PUT STR
0085 FOR I=0 TO LEN(A$)-1
0086 C=ASC(MID$(A$,I,1))
0087 GPUTCHR X,Y,"BGF0",C,9,2
0088 X=X+16:NEXT:RETURN
0089 @TITLE     '--------TITLE
0090 GPAGE 0:GCLS:L=0:PAL=8:LV=1
0091 C=0:W=64:H=64:GOSUB @BGFILL
0092 BGOFS L,0,0:A$="ﾊﾟﾜｰﾝﾅｻﾞ"
0093 @TWAIT     '----TOUCH WAIT
0094 GPAGE 1:GCLS 240
0095 X=48:Y=48:GOSUB @PUTSTR
0096 X=32:Y=132:A$="(TOUCH HERE)"
0097 GOSUB @PUTSTR
0098 @TCH2
0099 IF TCHST==0 THEN @TCH2
0100 BEEP 7:BGMSTOP:RETURN
```

Background fill.



▶ Set the speed and angle using the lower screen, and let your bomb fly. If you can get your enemies caught in the bomb blast, you will proceed to the next stage.

Check Point

It's not the most original idea in the world, but at least it's a proper game.

I'll bet you could use that ATAN function for calculating angles in lots of other games.

I think the problem here is that the program uses lots of colons, which can make it hard to read. But I suppose that can't be helped when you've only got 100 lines.

Extra!

Give these a try and have even more fun with Petit Computer

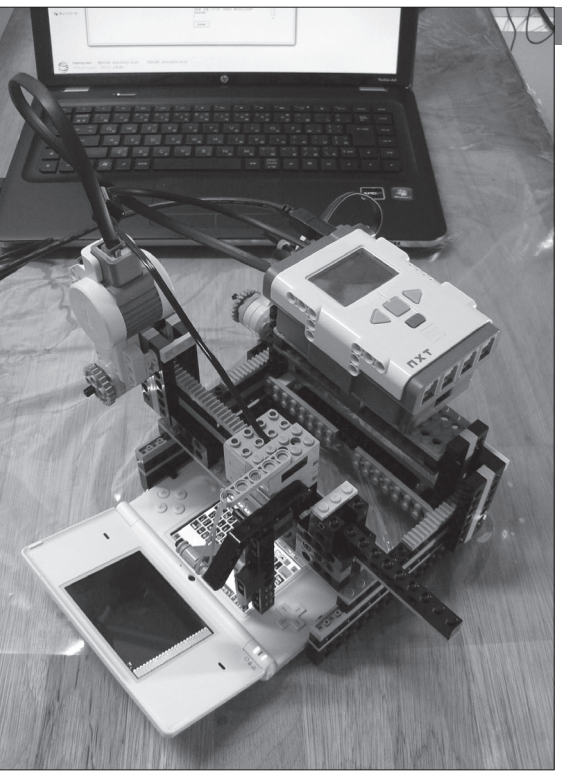
New Twists on Petit Computer

●Using a Magnifying Glass

Programming using the Nintendo DSi for long periods of time can lead to tired eyes. This is why some bright spark suggested using a magnifying glass with a 12cm diameter with its own stand. This doubles the size of the screen display, but it does mean that you can't move your head while you program...

●Using Your Thumbs

This is a method suggested by Hiroshi Yamasaki, who we met in the Single-Screen Programming Corner on page 68. Instead of using the stylus, he typed programs with his thumbs. The use of both hands increases efficiency, but the size of your thumbs means this method can only be used on the DSi XL or 3DS XL. It can also leave thumb prints on the screen...



● Using a Robot

And here is the ultimate weapon that we've been dying to introduce. We created a robot with the ability to type using the building blocks in the LEGO Mindstorms NXT kit. The motor makes the robot move the stylus along the x, y and z axes and enter commands via the Petit Computer keyboard (data is sent via USB). In our trial, it took the robot about an hour to write a 160-line program with only 5 errors.

■ To see this programming robot in action in a Japanese video clip, check out this link:
http://www.youtube.com/watch?v=KuZa-361hPo

PART5

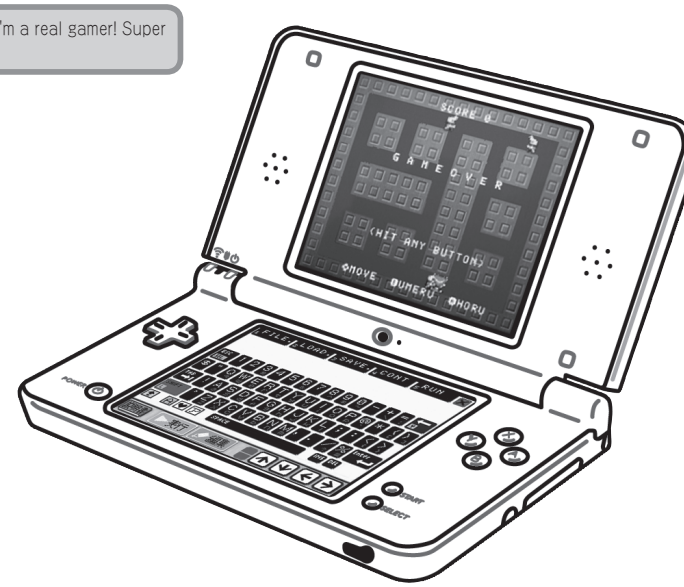
The Epic Program Challenge

This section introduces the ultimate goal of this entire guide. The aim is to make use of the Petit Computer BASIC techniques you have learned and to create a large-scale program of your very own. We can't wait to see what you'll come up with. It's time to really put Petit Computer through its paces.

So there are two programs here which are about 300 lines long. Interesting...

The maximum number of lines is actually 9999. But I think these will be enough to be getting on with. It's hard work to enter all these lines, after all!

'Whoa! I'm a real gamer! Super Nova!'



※Super Nova was the special move used by the eponymous hero of the hit Japanese manga series 'Game Center Arashi'. This move was so powerful that it actually put Arashi's life in danger.

5-01 3D Effect Shooting Game

Before polygon graphics became common in video games, BASIC used graphic scaling to produce a pseudo-3D effect in a lot of games. With Petit Computer, you can use sprites to make graphics with a more modern feel, but for a little blast of nostalgia from the past, we tried to come up with an old-style shooter.

The Epic Program No.1

Be Fast and Accurate and Hit those Space Ships

SPACE MOLE

● Introducing the Program

This is a pseudo-3D shooting game set in the depths of space. The graphics make use of GLINE commands to give the game a real retro feel.

● How it Works

Your aim is to hit the red triangular alien craft with your missiles and wipe them out. Touching or sliding the stylus across the Touch Screen will move the target, with the A, B, X, Y Buttons and the +Control Pad all firing missiles. You have a limited supply of ammunition, so try not to waste it. When you have taken out a certain number of foes, you will progress to the next level. When enemies strike you, your shield will be depleted, and when it reaches zero, it's game over.

Program List

```
0001 '-----------SPACE MOLE
0002 CLEAR
0003 DIM B(20)
0004 DIM X1(20),Y1(20)
0005 DIM GX(20),GY(20)
0006 DIM GA(20),GZ(20)
0007 DIM M(10),N(10)
0008 DIM M1(10),N1(10)
0009 BMX=90:ZMX=48:Z1=1:HMX=5
0010 '--------------RETRY
0011 @START
0012 GOSUB @GINIT
0013 GOSUB @LVINIT
```

←The radius of the explosions(variable BMX).The maximum distance to enemies(variable ZMX).The setting for the number of stars (variables HMX).

Title screen.

✚ To Next Page ✚

```
0014 Y=8:A$="S P A C E"
0015 GOSUB @MES
0016 Y=12:A$="M O L E"
0017 GOSUB @MES
0018 BGMPLAY 1
0019 GOSUB @TWAIT
0020 @START2
0021 GOSUB @GINIT
0022 GOSUB @DATINIT
0023 BGMSTOP
0024 CLS
0025 Y=10
0026 A$="R E A D Y"
0027 GOSUB @MES
0028 Y=15
0029 A$="S T A G E  "+STR$(LV)
0030 GOSUB @MES
0031 J=120:GOSUB @CWAIT
0032 CLS
0033 GOSUB @PUTENE
0034 BGMPLAY 17
0035 '---------------MAIN
0036 @MAIN                              ←Main loop.
0037 GCLS
0038 F=0
0039 FOR I=0 TO 19
0040 IF B(I)<0 THEN @ENDFOR
0041 F=F+1
0042 X=GX(I):Y=GY(I)
0043 IF B(I)==0 THEN @MOVE
0044 B(I)=B(I)+3                        ┐
0045 GCIRCLE X,Y,B(I),12                ├Operation during explosions. Draws circle using
0046 IF B(I)<BMX THEN @ENDFOR           ┘GCIRCLE command.
0047 B(I)=-1
0048 GOTO @ENDFOR
0049 '-------------MOVE
0050 @MOVE                              ┐
0051 IF I<10 THEN GOSUB @TEKIMOVE       ├Movement control.
0052 IF I>=10 THEN GOSUB @MYMOVE        ┘
0053 IF GZ(I)>=ZMX THEN @DAMAGE         ┐
0054 GOTO @ENDFOR                       │
0055 @DAMAGE                            │
0056 B(I)=-1                            ├Operation when your missiles or enemies reach
0057 IF I>9 THEN @ENDFOR                │target.
0058 S=S-25                             │
0059 GCLS 2                             ┘
```

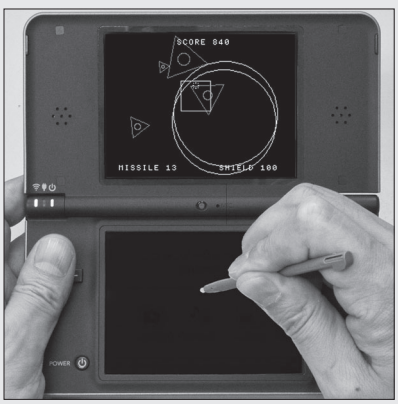
✚ To Next Page ✚

```
0060 GOSUB @PUTENE
0061 BEEP 11
0062 @ENDFOR
0063 NEXT
0064 '-----------------TEKI
0065 GOSUB @TEKIFIRE                ←Enemy attacks.
0066 GOSUB @FIRE                    ←Player attacks.
0067 IF S<=0 THEN @GAMEOVER
0068 IF (F+TE)==0  THEN @NEXTLV
0069 GOSUB @HOSHI
0070 VSYNC 1
0071 GOTO @MAIN
0072 '--------------------
0073 @MV
0074 GZ(I)=GZ(I)+Z1
0075 GX(I)=GX(I)+X1(I)
0076 GY(I)=GY(I)+Y1(I)
0077 RETURN
0078 '------------
0079 @MYMOVE
0080 GOSUB @MV
0081 X=GX(I)
0082 Y=GY(I)
0083 Z=ZMX-GZ(I)
0084 L=Z/2
0085 GBOX X-L,Y-L,X+L,Y+L,11        ←Display bullets with GBOX command.
0086 J=0
0087 @HITCHK                        ←Collision detection for missiles.
0088 IF B(J) THEN @HSKIP
0089 BX=GX(J):BY=GY(J):BZ=GZ(J)
0090 IF ABS(Z-BZ)>8 THEN @HSKIP
0091 DX=ABS(X-BX)
0092 IF DX>L THEN @HSKIP
0093 DY=ABS(Y-BY)
0094 IF DY>L THEN @HSKIP
0095 B(J)=1:B(I)=1:P=0              ←Sets explosion flag when missile hits target (array
0096 L=L/2                          variable B).
0097 IF DX<L AND DY<L THEN P=100    ←High score when hitting a narrower target.
0098 BEEP 13,P*40
0099 SC=SC+P+10
0100 GOSUB @PUTENE
0101 RETURN
0102 @HSKIP
0103 J=J+1
0104 IF J<10 THEN @HITCHK
0105 RETURN
```

✚ To Next Page ✚

```
'--------------------
@TEKIMOVE
GA(I)=(GA(I)+1)%360
IF RND(9)<LV THEN GOSUB @MV
X=GX(I)
Y=GY(I)
Z=GZ(I)
A=GA(I)
TX1=COS(RAD((A+  0)%360))*Z+X
TY1=SIN(RAD((A+  0)%360))*Z+Y
TX2=COS(RAD((A+120)%360))*Z+X
TY2=SIN(RAD((A+120)%360))*Z+Y
TX3=COS(RAD((A+240)%360))*Z+X
TY3=SIN(RAD((A+240)%360))*Z+Y
GLINE TX1,TY1,TX2,TY2,2
GLINE TX3,TY3,TX2,TY2,2
GLINE TX1,TY1,TX3,TY3,2
GCIRCLE X,Y,Z/4 ,2
RETURN
'---------------GAME OVER
@GAMEOVER
BGMPLAY 15
Y=8
A$="G A M E  O V E R"
GOSUB @MES
GOSUB @PUTENE
J=150:GOSUB @CWAIT
GOSUB @TWAIT
GOTO @START
'---------------CLEAR
@NEXTLV
BGMSTOP
Y=10
A$="S T A G E  C L E A R"
GOSUB @MES
BEEP 7
VSYNC 10
J=120:GOSUB @CWAIT
LV=LV+1
GOTO @START2
'---------------PUT CURSOR
@CURSOR
AX=TCHX:AY=TCHY
SPOFS 0,AX,AY
BTN=BUTTON()
RETURN
```

Enemy movement and display



▲Target matches stylus movement. A, B, X, Y Buttons and +Control Pad launch missiles.

✚ To Next Page ✚

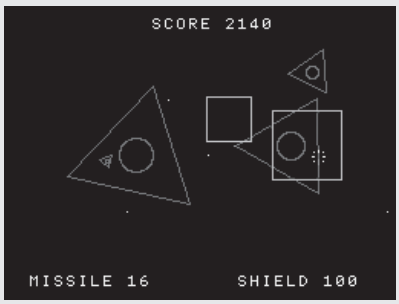
```
'---------------MOVE HOSHI
@HOSHI
FOR I=0 TO HMX-1
X=M(I)+M1(I):Y=N(I)+N1(I)
IF X<1 OR X>254 THEN @HOSHI1
IF Y<1 OR Y>191 THEN @HOSHI1
GPSET X,Y,15
M(I)=X:N(I)=Y
GOTO @HOSHI2
@HOSHI1
M(I)=128:N(I)=96
R=(RND(10)/5)+0.5
T=RAD(RND(360))
M1(I)=COS(T)*R
N1(I)=SIN(T)*R
@HOSHI2
NEXT
RETURN
'---------------WAIT CURSOR
@CWAIT
GCLS
GOSUB @HOSHI
GOSUB @CURSOR
VSYNC 1
J=J-1:IF J THEN @CWAIT
RETURN
'---------------MY MISSILE
@FIRE
GOSUB @CURSOR
FCNT=FCNT+1
IF FCNT<16 THEN RETURN
FCNT=16
IF BTN==0 THEN RETURN
FCNT=0
IF E<=0 THEN BEEP 9:RETURN
I=10
@FIRE1
IF B(I)<0 THEN @FIRE2
I=I+1
IF I>19 THEN RETURN
GOTO @FIRE1
@FIRE2
E=E-1
BEEP 10
BX=AX
BY=AY
```

Movement of stars in background.

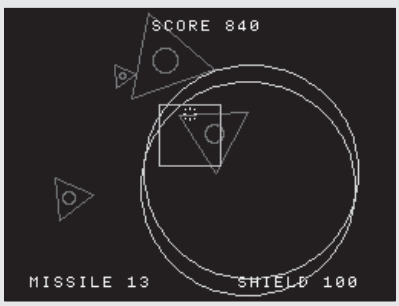
Button input, missile launch.

+ To Next Page +

```
0198 GOSUB @PUTENE
0199 @SETXY
0200 GX(I)=BX
0201 GY(I)=BY
0202 X1(I)=(AX-BX)/(ZMX/Z1)
0203 Y1(I)=(AY-BY)/(ZMX/Z1)
0204 B(I)=0
0205 GZ(I)=0
0206 RETURN
0207 '--------------TEKI MISSILE
0208 @TEKIFIRE
0209 IF TE<=0 THEN RETURN
0210 IF RND(50) THEN RETURN
0211 I=0
0212 @NEWB1
0213 IF B(I)<0 THEN @NEWB2
0214 I=I+1
0215 IF I>9 THEN RETURN
0216 GOTO @NEWB1
0217 '--------------SELECT TARGET
0218 @NEWB2
0219 AX=RND(256-64)+32
0220 AY=RND(192-48)+24
0221 TE=TE-1
0222 BX=RND(256)
0223 BY=RND(192)
0224 GOTO @SETXY
0225 '---------------INIT SCREEN
0226 @GINIT
0227 CLS
0228 SPPAGE 0
0229 SPCLR
0230 SPSET 0,255,2,0,0,2
0231 PNLTYPE "OFF"
0232 GPAGE 0
0233 GCLS
0234 RETURN
0235 '---------------INIT SCORE
0236 @LVINIT
0237 SC=0
0238 LV=1
0239 @DATINIT
0240 F=0
0241 TE=12
0242 FOR I=0 TO 19
0243 B(I)=-1
```



▲Blow the red triangular alien craft out of the skies with your missiles. Each time you clear a stage, your enemies' speed will increase.



▲Hitting an enemy will result in a huge explosion. If you hit an enemy right in the center, you will get maximum points.

✚ To Next Page ✚

```
0244 NEXT
0245 E=20
0246 S=100
0247 RETURN
0248 '---------------PRINT ENERGY
0249 @PUTENE
0250 LOCATE 2,22
0251 PRINT "MISSILE ";E;" "
0252 LOCATE 19,22
0253 PRINT "SHIELD ";S;" "
0254 LOCATE 12,1
0255 PRINT "SCORE ";SC
0256 RETURN
0257 '---------------WAIT
0258 @TWAIT
0259 IF TCHST THEN @TWAIT
0260 GPAGE 1
0261 GCLS 14
0262 PNLSTR 9,10,"TOUCH SCREEN",2
0263 @TOUCH2
0264 VSYNC 1
0265 IF TCHST==0 THEN @TOUCH2
0266 GCLS 14
0267 GPAGE 0
0268 RETURN
0269 '--------------PRINT MESSAGE
0270 @MES
0271 LOCATE 16-(LEN(A$)/2),Y
0272 PRINT A$;
0273 RETURN
```

▲Get hit by too much enemy firepower and your shield will hit zero and it will be game over.

Check Point

So using flat shapes in this way is a way of creating a pseudo-3D effect.

But those alien craft look pretty flimsy to me!

This program makes use of the Touch Screen, meaning that it's making the most of Petit Computer's unique features.

5-02 Maze Action Game

In the 1970s, at the dawn of the video game era, there were many games which, while they did not come close to today's games in terms of graphics, boasted great ideas which made them classics we remember fondly to this day. We have used the more advanced graphical capabilities of Petit Computer to come up with our version of a classic maze game.

The Epic Program No.2

A Maze Game that Respects the Traditions of Times Past

Heisei Alien

●Introducing the Program

This game reboots the early Japanese classic, Heiankyo Alien.

●How it Works

You control the player character with the +Control Pad, dig holes in the ground by pressing the A Button, and fill them in again with the B Button. You clear each stage by trapping and burying all the aliens.

Once an alien falls into a hole, there is a limited amount of time before they climb out. If a hole is not big enough, it will be refilled if an alien passes over it.

Program List

```
0001 '------------HEISEI ALIEN
0002 CLEAR
0003 DIM Q(15,12)
0004 DIM TX(10),TY(10)
0005 DIM TD(10),TM(10)
0006 DIM X1(4),Y1(4)
0007 X1(0)= 1:Y1(0)= 0
0008 X1(1)= 0:Y1(1)= 1
0009 X1(2)=-1:Y1(2)= 0
0010 X1(3)= 0:Y1(3)=-1
0011 QF=5                    ←Constant assigned to variable QF when there is a hole in the ground.
0012 '------------RETRY
0013 @RETRY
```

+ To Next Page +

```
0014 GOSUB @SCRINIT
0015 CY=9:A$="H E I S E I"
0016 GOSUB @PUTA
0017 CY=11:A$="A L I E N"
0018 GOSUB @PUTA
0019 LV=1
0020 BGMPLAY 10
0021 GOSUB @HITANY
0022 '------------START
0023 @START
0024 GOSUB @SCRINIT
0025 GOSUB @MAPINIT
0026 GOSUB @MYINIT
0027 GOSUB @TEKIINIT
0028 CLS
0029 BGMSTOP
0030 CY=8
0031 A$="S T A G E "+STR$(LV)
0032 GOSUB @PUTA
0033 CY=12:A$="R E A D Y"
0034 GOSUB @PUTA
0035 VSYNC 180
0036 CLS
0037 BGMPLAY 26
0038 CY=22
0039 A$="✥MOVE  ❽UMERU  ❼HORU"
0040 GOSUB @PUTA
0041 GOSUB @PUTSC
0042 OVR=0
0043 '------------MAIN LOOP
0044 @MAIN
0045 VSYNC 1
0046 GOSUB @MYCTR
0047 GOSUB @TEKIMOVE
0048 IF TCT==0 THEN @NEXTLV
0049 IF OVR==1 THEN @GAMEOVER
0050 GOTO @MAIN
0051 '-----------CONTROL MY
0052 @MYCTR
0053 IF M THEN @MY2
0054 BTN=BUTTON()
0055 CNT=(CNT+1)%10
0056 IF CNT THEN BTN=BTN AND &H0F
0057 IF BTN==0 THEN RETURN
0058 IF BTN AND 1 THEN D=3
0059 IF BTN AND 2 THEN D=1
0060 IF BTN AND 4 THEN D=2
```

Title screen display.

←Starts main loop.

←9 times out of every 10 loops, the A/B Buttons will be disabled.

Explain

"Heian Ailien"

This is Maze Action Game developed by members of Tokyo University. This was developed for PC(Apple2) Game in 1979, it was ported into arcade game by the manufacturer in next year. Its peculiarity is how we fight against Aliens.It is by digging and burying them.

There are a lot of games ported across their machines.

We get the permission of this game from the member of this game developer team "Arimasa Takeshige".

✚ To Next Page ✚

```
0061 IF BTN AND 8 THEN D=0
0062 IF BTN AND &H50 THEN @HORU
0063 IF BTN AND &HA0 THEN @UME
0064 SPCHR 0,(D*4)+64
0065 GOSUB @BGCHKMY
0066 IF C THEN RETURN
0067 BX=BX/2
0068 BY=BY/2
0069 IF Q(BX,BY) THEN RETURN
0070 M=16
0071 '---------MOVE MY
0072 @MY2
0073 M=M-1
0074 X=X+X1(D):Y=Y+Y1(D)
0075 SPOFS 0,X,Y
0076 RETURN
0077 '---------HORU
0078 @HORU
0079 J=1
0080 GOTO @UME2
0081 '---------UMERU
0082 @UME
0083 J=-1
0084 @UME2
0085 GOSUB @BGCHKMY
0086 IF C THEN RETURN
0087 BX=BX/2
0088 BY=BY/2
0089 I=Q(BX,BY) + J
0090 IF I>QF THEN RETURN
0091 IF I<0 THEN RETURN
0092 BEEP 9
0093 Q(BX,BY)=I
0094 GFILL CX,CY,CX+15,CY+15,1
0095 IF I==0 THEN @UMETEKI
0096 CX=CX+8:CY=CY+8
0097 GCIRCLE CX,CY,I+2,13
0098 GPAINT CX,CY,14
0099 IF I==QF THEN BEEP 8
0100 RETURN
0101 '-----------TEKI UMERU
0102 @UMETEKI
0103 FOR I=0 TO 9
0104 IF CX!=TX(I) THEN @ENDFOR2
0105 IF CY!=TY(I) THEN @ENDFOR2
0106 TY(I)=-1
0107 SC=SC+10
```

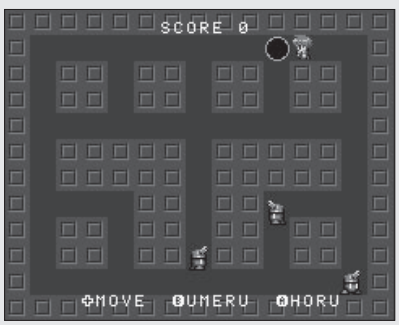
←The player character display will switch depending on the direction it is moving.

←When there is a hole in front of the PC, it will be unable to move forward.

←Sets the number of pixels the PC moves with each step.

←PC movement.

←Digging and filling holes. The array Q will store the current state of each hole.



▲The player character can dig holes and then refill them.

✚ To Next Page ✚

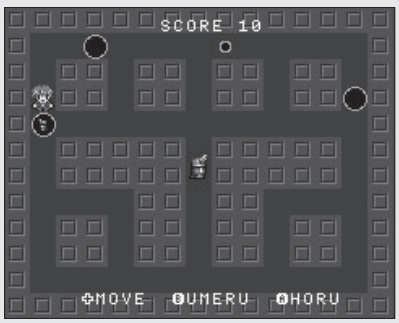
```
0108 BEEP 13
0109 @ENDFOR2
0110 NEXT
0111 GOSUB @PUTSC
0112 RETURN
0113 '-----------INIT MY CHR
0114 @MYINIT
0115 PAL=2
0116 N=0:D=1:C=68:M=0
0117 X=7*16:Y=5*16
0118 SPSET N,C,PAL,0,0,2
0119 SPOFS N,X,Y
0120 SPANIM N,4,10
0121 RETURN
0122 '-----------MOVE TEKI
0123 @TEKIMOVE                          ←Enemy movement.
0124 TCT=0
0125 FOR I=0 TO TMX-1
0126 N=I+1
0127 CX=TX(I):CY=TY(I)
0128 IF CY<0 THEN @TEKICLR
0129 TCT=TCT+1
0130 IF TD(I)>3 THEN @ANA2
0131 IF TM(I) THEN @TEKI2
0132 J=TD(I)                            ┐
0133 J=J+RND(3)-1                       │
0134 IF J>=4 THEN J=J-4                 ├Enemy direction of movement switched using random
0135 IF J<0  THEN J=J+4                 │values.
0136 CX=CX+(X1(J)*16)                   ┘
0137 CY=CY+(Y1(J)*16)
0138 GOSUB @BGCHK
0139 IF C THEN @ENDFOR
0140 TD(I)=J
0141 TM(I)=16                           ←Sets the number of pixels enemies move with each step.
0142 C=(LV%2)*16+128+(J*2)
0143 SPCHR N,C
0144 GOTO @ENDFOR
0145 @TEKI2
0146 J=TD(I)
0147 CX=CX+X1(J):CY=CY+Y1(J)
0148 SPOFS N,CX,CY
0149 TX(I)=CX:TY(I)=CY
0150 TM(I)=TM(I)-1
0151 IF TM(I)%16 THEN @HITCHK
0152 BX=CX/16:BY=CY/16
0153 IF Q(BX,BY) THEN @ANAIN
0154 @HITCHK
```

+ To Next Page +

```
0155 IF ABS(CX-X)>8 THEN @ENDFOR
0156 IF ABS(CY-Y)>8 THEN @ENDFOR
0157 OVR=1
0158 GOTO @ENDFOR
0159 '------------ANA OCHIRU
0160 @ANAIN
0161 TM(I)=0
0162 IF Q(BX,BY)<QF THEN @ANAOUT
0163 SPOFS N,CX+4,CY+4
0164 SPSCALE N,50
0165 TD(I)=NTM
0166 BEEP 6
0167 GOTO @ENDFOR
0168 '------------ANA DERU
0169 @ANA2
0170 BX=CX/16:BY=CY/16
0171 IF Q(BX,BY)==0 THEN @ANAOUT
0172 TD(I)=TD(I)-1
0173 IF TD(I)>3 THEN @ENDFOR
0174 @ANAOUT
0175 TD(I)=RND(4)
0176 SPSCALE N,100
0177 BEEP 1
0178 Q(BX,BY)=0
0179 BX=BX*16:BY=BY*16
0180 GFILL BX,BY,BX+15,BY+15,1
0181 GOTO @ENDFOR
0182 '------------CLEAR TEKI
0183 @TEKICLR
0184 SPOFS N,-16,-16
0185 @ENDFOR
0186 NEXT
0187 RETURN
0188 '------------INIT TEKI
0189 @TEKIINIT
0190 TMX=LV+2
0191 IF TMX>10 THEN TMX=10
0192 PAL=3
0193 FOR I=0 TO 9
0194 CY=-16
0195 IF I>=TMX THEN @INI3
0196 @INI2
0197 CX=RND(5)*48+16
0198 CY=RND(4)*48+16
0199 IF ABS(CX-X)<32 THEN @INI2
0200 IF ABS(CY-Y)<32 THEN @INI2
0201 @INI3
```

←When the PC and enemies come into contact, the variable OVR which stands for the Game Over flag is set for 1.

←When the array TD is greater than 3, an enemy is in a hole.



▲Fill in holes where enemies are trapped to defeat them.

←Sets maximum number of enemies.

✚ To Next Page ✚

```
0E02 TX(I)=CX:TY(I)=CY
0E03 TD(I)=RND(4)
0E04 TM(I)=0
0E05 N=I+1
0E06 C=(LV%2)*16+128
0E07 SPSET N,C,PAL,0,0,2
0E08 SPOFS N,CX,CY
0E09 SPANIM N,2,10
0E10 NEXT
0E11 NTM=260-(LV*10)
0E12 IF NTM<140 THEN NTM=1
0E13 RETURN
0E14 '------------INIT SCREEN
0E15 @SCRINIT
0E16 CLS
0E17 SPPAGE 0
0E18 SPCLR
0E19 GPAGE 0
0E20 GCLS
0E21 BGPAGE 0
0E22 @BGCLR
0E23 PAL=0
0E24 FOR LAY=0 TO 1
0E25 W=32:H=24:C=0:X=0:Y=0
0E26 GOSUB @BGPS
0E27 BGOFS LAY,0,0
0E28 NEXT
0E29 RETURN
0E30 '------------INIT MAP          ←Resets maze.
0E31 @MAPINIT
0E32 CLS
0E33 GFILL 0,0,30*8,24*8,1
0E34 FOR Y=0 TO 11
0E35 FOR X=0 TO 14
0E36 Q(X,Y)=0
0E37 NEXT
0E38 NEXT
0E39 PAL=9
0E40 LAY=0
0E41 W=2:H=2:C=30
0E42 FOR X=0 TO 28 STEP 2
0E43 Y=0:GOSUB @BGPS
0E44 Y=22:GOSUB @BGPS
0E45 NEXT
0E46 FOR Y=0 TO 22 STEP 2
0E47 X=0:GOSUB @BGPS
0E48 X=28:GOSUB @BGPS
```

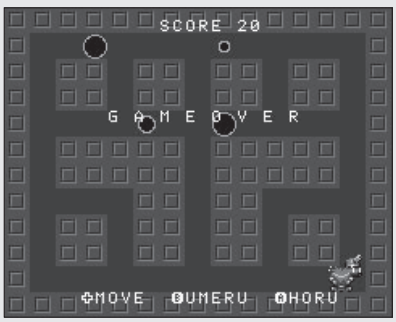
+ To Next Page +

```
0249 NEXT
0250 FOR BY=0 TO 16 STEP 2
0251 IF (BY%6)==0 THEN @MAP2
0252 FOR BX=0 TO 22 STEP 2
0253 IF (BX%6)==0 THEN @MAP3
0254 X=BX+2:Y=BY+2
0255 GOSUB @BGPS
0256 @MAP2
0257 NEXT
0258 @MAP3
0259 NEXT
0260 FOR BY=8 TO 14 STEP 6
0261 FOR BX=8 TO 20 STEP 12
0262 X=BX:Y=BY
0263 D=RND(4)
0264 FOR M=0 TO 1
0265 X=X+X1(D)*2:Y=Y+Y1(D)*2
0266 GOSUB @BGPS
0267 NEXT
0268 NEXT
0269 NEXT
0270 RETURN
0271 '-------------PUT BG
0272 @BGPS
0273 FOR I=0 TO H-1
0274 FOR J=0 TO W-1
0275 CC=C+(I*32)+J
0276 IF C<16 THEN CC=C
0277 BGPUT LAY,X+J,Y+I,CC,PAL,0,0
0278 NEXT
0279 NEXT
0280 RETURN
0281 '-------------CHECK MY
0282 @BGCHKMY
0283 CX=X+X1(D)*16
0284 CY=Y+Y1(D)*16
0285 '-------------CHECK BG
0286 @BGCHK
0287 BX=FLOOR(CX/8)%64
0288 BY=FLOOR(CY/8)%64
0289 BGREAD(LAY,BX,BY),C,PAL,H,V
0290 RETURN
0291 '-------------PRINT SCORE
0292 @PUTSC
0293 CY=1
0294 A$= "SCORE "+STR$(SC)
0295 '-------------PRINT TEXT
```

Blocks a number of the maze's paths.

+ To Next Page +

```
@PUTA
LOCATE 16-LEN(A$)/2,CY
PRINT A$;
RETURN
'------------NEXT LEVEL
@NEXTLV
BGMPLAY 9
CY=8
A$="S T A G E C L E A R"
GOSUB @PUTA
LV=LV+1
VSYNC 5*60
GOTO @START
'------------GAME OVER
@GAMEOVER
BGMPLAY 4
CY=8
A$="G A M E O V E R"
GOSUB @PUTA
SPCHR 0,88
VSYNC 180
GOSUB @HITANY
GOTO @RETRY
'------------HIT ANY KEY
@HITANY
IF BUTTON() THEN @HITANY
CY=17
A$="(HIT ANY BUTTON)"
GOSUB @PUTA
@HITANY2
VSYNC 1
IF BUTTON()==0 THEN @HITANY2
RETURN
```



▲If you bump into an alien, it's game over. Be sure to plan where to dig holes carefully.

Check Point

This is really old-fashioned. Maybe it's because it uses sprites, but it doesn't have the feel of the original.

If you remade it using character strings to create the graphics, it might feel more faithful to the original.

With the capabilities of the Petit Computer, it wouldn't be a problem to make the background scroll. It would be good to build on this and come up with a totally different game.

Appendix

Petit Computer Resources

There is a detailed set of instructions included in the Petit Computer software which can be viewed by pressing the Help button on the lower screen keyboard. The same information is also available on the Petit Computer website.

It can be a little fiddly switching between Edit Mode and the manual when you are working on a program. Some users may prefer the ease of studying the instructions online while programming on their Nintendo DSi or 3DS.

※Color palette, character code, system icon, BG/sprite character, they are in page 10-16.

Appendix 1 System Variables

Numeric System Variables

	Read	Write	
CSRX	○	×	Current cursor x-axis (horizontal) position
CSRY	○	×	Current cursor y-axis (vertical) position
FREEMEM	○	×	Remaining memory available to user(Kbyte)
VERSION	○	×	System Version (0xAABBCCDD, Version AA.BB.CC.DD)
ERR	○	×	Error number immediately after error occurred
ERL	○	×	Number of line where error occurred.
RESULT	○	×	x coordinate pressed on Touch Screen.
TCHX	○	×	y coordinate pressed on Touch Screen.
TCHY	○	×	Touch status(TRUE = Touched)
TCHST	○	×	Time Touch Screen is touched for(Given in number of frames)
TCHTIME	○	×	Frames elapsed since start of program (max. 145 minutes)
MAINCNTL	○	×	Duration of frame display since program launched (data over 145 minutes)
MAINCNTH	○	×	Amount TAB key will move(0-16)
TABSTEP	○	○	Always 1
TRUE	○	×	Always 0
FALSE	○	×	Always -1
CANCEL	○	×	"FALSE=Don't use
ICONPUSE	○	○	TRUE=Use"
ICONPAGE	○	○	Page number for user system icon(0 is always entered in Run Mode)
ICONPMAX	○	○	Maximum number of pages for user system icon(Does not work in Run Mode.)
FUNCNO	○	×	Number of function key pressed (1-5, 0=not pressed)
FREEVAR	○	×	Number of variables that can be saved
SYSBEEP	○	○	System sound effect controls (True = ON, False = OFF)

Text String System Variables

	Read	Write	
TIME$	○	×	Obtains current time as a string (HH:MM:SS)
DATE$	○	×	Obtains current date as a string (YYYY/MM/DD)
MEM$	○	○	Number of strings that can be saved in file(MAX 255characters)

Appendix 2 Error Number Chart

When you get Error, Error Number is in System Variable ERR, and Line Number is in ERL.

1	Syntax error	There is problematic grammar in the program.
2	Out of range	The value exceeds the valid range.
3	Out of memory	There is insufficient memory available.
4	Undefined label	The destination for a branch instruction cannot be located.
5	NEXT without FOR	There is a NEXT command which does not belong to any FOR command.
6	Out of DATA	There is insufficient DATA available for a READ command.
7	Illegal function call	There is a problem with the assignment of elements in a function or command.
8	Duplicate definition	The same array or variable has been defined more than once.
9	Can't continue	A program cannot be continued using a CONT command.
10	Missing operand	There are insufficient parameters.
11	Duplicate label	The same label has been defined more than once.
12	Illegal resource type	The resource type designated by a string does not exist.
13	Illegal character type	The designated character type does not exist.
14	String too long	The string is too long. Labels should be no longer than 8 characters, while strings should be no more than 256 characters in length.
15	Division by zero	A number has been divided by zero.
16	Overflow	The results of an operation have exceeded the permitted range.
17	Subscript out of range	The subscript for an array variable is out of range.
18	Type mismatch	Variable types do not match.
19	Fomula too complex	The formula may have too many bracketed sections, or otherwise be too complex.
20	RETURN without GOSUB	A RETURN command is present without an accompanying GOSUB command.
21	FOR without NEXT	A FOR command is present which does not correspond to a NEXT command.

Appendix 3 Commands List

01 Run Mode-Specific Commands NEW、LIST、RUN、CONT、FILES

There are 5 commands that can only be used in Run Mode. These are related to running and continuing programs and cannot be written into the programs themselves:

NEW

This deletes the program.

Format	NEW
Parameters	None

LIST

Switches to Edit Mode and begins edit.

Format	LIST	
	LIST @label	
	LIST line number	
Parameters	Line number	Designate line where source is displayed(Can Be Omitted).
	@label	Designate line where source is displayed(Can Be Omitted).
Error		When line number or label does not exist.

RUN

This runs the program.

Format	RUN
Parameters	None

CONT

Continues a program stopped with a STOP command.

Format	CONT
Parameters	None
Error	When program has been run and cannot be continued.

FILES

Displays a list of files on the console screen.

Format	FILES [file type name [, file type name...]]	
Parameters	File Type Name	Designate if you want display only specified resource.

02 Basic Commands

CLEAR、=、DIM、REM、@、KEY、VSYNC、ON～GOTO、ON～GOSUB、GOTO、GOSUB、RETURN、STOP、END、FOR～TO～STEP、NEXT、IF～THEN、IF～GOTO、READ、DATA、RESTORE、TMREAD()、DTREAD()

Initialize memory,Conditional Branch ,Loop,Conditional Judgment,Read and Write data,etc・・・

CLEAR

This resets variable names and BASIC internal memory.

Format	CLEAR
Parameters	None

=(LET)

Assign (an abbreviation of the LET command)

Format	ABC=123
	TEXT$="ABCDE"
Parameters	None

DIM

Array declarations.element count for up to 2 dimensions. Element count can be defined up to a total of 32768.

Format	DIM pos(4),size(4)
	DIM sample(10, 5)
Parameters	None

'(REM)

For annotations(comments). The text following this command up to the next linebreak will be ignored.

Format	REM The following is a comment
	' comment text
Parameters	None

@

LABEL definition. This always needs to be added at the start of a row. It can be used to give destinations with commands such as GOTO and GOSUB. 8 character string(except space) following @ will become the name under which it is saved. When you use this insted of line number, you must write '@' the top of the line.

Format	@NAME1	
	@SAMPLE The end is after 8 characters or space, after that the text up to the next linebreak will be ignored.	
Parameters	Label name	Unlike strings, it is not necessary to enclose with ""
Error		When characters or symbols that are not alphanumeric characters or '_' are in a label name. When the label name is not defined.

SAMPLE : Back to the label line.

```
0001 A=0
0002 @1
0003 A=A+1
0004 PRINT A
0005 GOTO @1
```

KEY

Assigning Strings to Function Keys. When used in a user-created program, the string data assigned to the function key will be entered in the program when that key is pressed.

Format	KEY number, "string"	
Parameters	Number	Function key number(1-5)
	String	Only 4 characters from the assigned string will be displayed, but strings of up to 256 characters can be saved. When the full string cannot be displayed, the last character will be displayed as '.'
Error		When a number is designated that does not exist.

VSYNC

Same length as screen renewal time (waiting for graphics to be renewed).

※To use in loop, you can use wait and keep its process regularly.(1/60sec per 1frame)

Format	VSYNC Frame number	
Parameters	Frame number	Indicates number of frames since the VSYNC command immediately beforehand. (0 = ignore)

The example of use (P57 EXAMPLE 3-05,EXAMPLE 3-08,etc)

ON ~ GOTO

Causes program to branch depending on numeric values.

Format	Line number when ON variable GOTO variable =0 (or @label), numeric value 1, numeric value 2...
Parameters	None

SAMPLE : Processing with numerical values of the variable A.

```
0001 ON A GOTO @0,@1,@2
0002   OOOOOOOOOO  ←Operation with variable A NOT 0,1,2
0003 @0
0004   OOOOOOOOOO  ←Operation with variable A = 0
0005 @1
0006   OOOOOOOOOO  ←Operation with variable A = 1
0007 @2
0008   OOOOOOOOOO  ←Operation with variable A = 2
```

The example of use (P47 SAMPLE2)

ON ~ GOSUB

Calls a sub-routine based on number

Format	Line number (or @label) when ON variable GOSUB variable =0, numeric value 1, numeric value 2...
Parameters	None

The example of use (P77 Avoid Using GOTO Wherever Possible②)

GOTO

Forced branch.

Format	GOTO @label
Parameters	None

The example of use (P47 SAMPLE2,P55 EXAMPLE 3-03,etc)

GOSUB

Call sub-routine

Format	GOSUB @label
Parameters	None

The example of use (P65 Single-Screen Programming ①,P76 A Smart Approach to Programming,etc)

RETURN

Returns from a sub-routine.

Always use this command in conjunction with GOSUB.

Format	RETURN
Parameters	None
Error	If GOSUB is not done, display error message “RETURN without GOSUB”.

SAMPLE PROGRAM

```
0001 A=0
0002 GOSUB @SUB
0003 A=1
0004 GOSUB @SUB
0005 END
0006 @SUB
0007 PRINT A
0008 RETURN
```

STOP

Forces the currently running program to stop and returns to the console.

Program can be restarted by CONT.

Format	STOP
Parameters	None

END

Ends the program.

Format	END
Parameters	None

FOR ~ TO ~ STEP

Repeat the designated number of times (if STEP has been omitted, it will be treated as STEP1. If increase is added and the final value is less than the initial value. the FOR command will be skipped and the NEXT and subsequent commands will be run.)

Format	FOR variable = initial value TO final value [STEP increase]
Parameters	None

The example of use (P48 SAMPLE3,etc)

NEXT

End of FOR

Format	NEXT [variable name]
Parameters	None
Error	FOR command is not used.

IF ~ THEN

Conditional Judgment.
The IF command does not work across multiple lines.

Format	IF condition is met THEN command
	IF condition is met THEN @label
Parameters	None

SAMPLE PROGRAM

```
0001 IF A==5 THEN PRINT " A==5" :GOTO @1  ←Variable A is 5
0002 PRINT " A!=5"                        ←Variable A isn't 5
0003 @1
```

The example of use (P47 SAMPLE2,etc)

IF ~ GOTO

Conditional Judgment.
The IF command does not work across multiple lines.

Format	IF condition met GOTO @label
Parameters	None

READ

Reads DATA.

Format	READ obtained variable1 [, obtained variable2...]	
	READ A	
	READ B$	
	READ X,Y,Z,G$	
Parameters	Obtained variable	Variable storing information taken from DATA.Multiple designation possible.
Error	Read	When amount of data to be read is insufficient.

The example of use (P48 SAMPLE3,P61 EXAMPLE 3-10)

DATA

Definition of data to be read with READ command can include a mix of alphanumeric characters.

Format	DATA number, number	
	DATA "String","String"	
	DATA 123,"SAMPLE"	
Parameters	Data(alphanumeric characters)	Sequences of strings and numbers to be divided with ','

The example of use (P48 SAMPLE3,P61 EXAMPLE 3-10)

RESTORE

Changes position of DATA to be READ.

Format	RESTORE @label
Parameters	None

The example of use (P61 EXAMPLE 3-10)

TMREAD( )

Convert time string into number.

Format	TMREAD(“time string”), HOUR, MIN, SEC	
Parameters	Time string	HH:MM:SS Format of time string
	HOUR	Variable retrieving hour
	MIN	Variable retrieving minute
	SEC	Variable retrieving second

SAMPLE PROGRAM

```
TMREAD (TIME$) , H, M, S←H <== hour, M <== minute, S <== second
```

DTREAD( )

Converts the date string into a number.

Format	DTREAD(“date string”), YEAR, MON, DAY	
Parameters	Date string	Date displayed in format: YYYY/MM/DD
	YEAR	Variable retrieving year
	MON	Variable retrieving month
	DAY	Variable retrieving day

SAMPLE PROGRAM

```
DTREAD (DATE$) , Y, M, D←Y <== year, M <== month, D <== day
```

03 Basic Console Commands

CLS、COLOR、LOCATE、PRINT、CHKCHR()、BUTTON()、INKEY$()、INPUT、LINPUT

Displays characters on the console.

CLS

Erases the contents of the console screen.

Format	CLS
Parameters	None

The example of use (P54 EXAMPLE 3-01,P58 EXAMPLE 3-06,etc)

COLOR

Caractor coler is specified on console display.

Format	Color Palette number	
Parameters	Palette number	0～15 (Uses the 15th color of the 16 color palette assigned to BG screens)

SAMPLE PROGRAM:Change the Text Color "COLOR".

```
0001 FOR I=0 TO 15      ←Variable I : 0-15
0002 COLOR I            ←Set the palette number variable I.
0003 PRINT "COLOR";I    ←Display "COLOR" by palette number "I".
0004 NEXT
```

LOCATE

Designates the position of the character display on the console.

Format	LOCATE x coordinate, y coordinate	
Parameters	x coordinate	x coordinate(0-31) ※Out of useful range is not error
	y coordinate	y coordinate(0-23) ※Out of useful range is not error

The example of use (P46 SAMPLE1,P54 EXAMPLE 3-01,etc)

PRINT

Displays characters on the console.

Format	PRINT ""string"" PRINT variable PRINT variable$ PRINT variable;variable$;"string" PRINT ""string"",variable,variable $	
Parameters	;	Used when displaying multiple elements one after the other.
	,	When displaying multiple elements one after the other, adjustments are made for TAB position.

The example of use (P46 SAMPLE1,P76 A Smart Approach to Programming,etc)

CHKCHR()

Search for character numbers on the console.

Format	Variable = CHKCHR (x coordinate, y coordinate)	
Parameters	x coordinate	x coordinate(0-31) ※Out of useful range is not error
	y coordinate	y coordinate(0-23) ※Out of useful range is not error
Returns		0-255=character code(-1= outside range)

The example of use (P70 Single-Screen Programming Corner④,etc)

EXMPLE:Display "A" and get its character code.

```
LOCATE 15,11:PRINT "A"  ←Display "A" on the coordinate.
PRINT CHKCHR(15,11)     ←Display character code on the coordinate.
```

BUTTON()

This returns data from each button pressed.

Retrieves bitwise data on buttons being pressed simultaneously. For example, if up and right are pressed at the same time, it will return a value of 9.

Format	Variable=BUTTON()
Parameters	None
Returns	Number that corresponds to button.

1	Up on +Control Pad
2	Down on +Control Pad
4	Left on +Control Pad
8	Right on +Control Pad
16	A Button
32	B Button
64	X Button
128	Y Button
256	L Button
512	R Button
1024	START
2048	SELECT

The example of use (P55 EXMPLE 3-03,etc)

EXAMPLE:Display "UP" Up on +Control Pad is pressed.

```
IF (BUTTON() AND 1)==1 THEN PRINT "UP"
```

INKEY$()

Obtains a single character inputted on the keyboard.
When there is nothing inputted, it will return "".

Format	Variable$=INKEY$0	
Parameters	None	
Returns	Character variable	A single keyboard character will be returned.

The example of use (P56 EXMPLE 3-04,etc)

EXAMPLE : Wait till the keyboard touched.

```
0001 @1
0002 IF  INKEY$ () ==" "  THEN GOTO @1
```

INPUT

Obtain numbers or strings.
Wait for input by keyboard and put what is input into vaiable.

Format	INPUT ""string""; received variable	
	INPUT ""string""; received character variable$	
	INPUT ""string""; received variable, received character variable$	
Parameters	String	Explanatory text for entry(Can Be Omitted).
	Received variable	Text string variable or value for obtaining data entered on the keyboard.
	,	Use commas to split up commands, so you can enter multiple commands.

EXAMPLE:Display the inputted character;

```
0001 INPUT  A$   ←Store what is inputted into variable A$.
0002 PRINT  A$   ←Display the content of variable A$.
```

The example of use (P47 SAMPLE2, P54 EXMPLE 3-02,etc)

LINPUT

Retrieves string, including characters like ',' which cannot be entered via INPUT.

Format	LINPUT ["string";] received variable$	
Parameters	String	Explanatory text for entry(Can Be Omitted)
	Received variable	Variable for receiving a one line string entered via keyboard.

04 File & Communication Commands

LOAD、SAVE、DELETE、EXEC、RENAME、RECVFILE、SENDFILE

The following commands are used for operations relating to files, such as loading, saving and deleting.

LOAD

Loads file.

Format	LOAD "resource name:file name" [, display control]	
Parameters	Resource Name. Strings assigned to the resources to be read.	
	PRG	Program (Can be omitted)
	MEM	Memory
	COL0～COL2	color (0=BG, 1=SPRITE, 2=GRP)
	GRP0～GRP1	Graphics (0=upper,1=lower)
	SCU0～SCU1	User Screen(0=Foreground layer 1=Background layer)
	BGU0～BGU3	User's BG Characters
	SPU0～SPU7	User Sprite Characters
	Display control	Enter FALSE and the dialog box will not be displayed during loading.
Error	RESULT	FALSE　TRUE　CANCEL

SAVE

This saves a file. (A dialog box confirming the decision will appear.)

Format	SAVE "Resource name:file name"	
Parameters	Resource name	*Refer to LOAD
Error	RESULT	FALSE
		TRUE
		CANCEL

DELETE

Erases file. (A dialog box confirming the decision will appear.)

Format	DELETE "Resource name:file name"	
Parameters	Resource name	*Refer to LOAD
Error	RESULT	FALSE
		TRUE
		CANCEL

EXEC

Loads and runs other programs from within the current program.

Format	EXEC "file name"	
Parameters	File name	Name of program file to run
Error	RESULT	FALSE = Failure

RENAME

Changes file names.

Format	RENAME "resource name: file name", "new name"	
Parameters	Resource name	*Refer to LOAD
Error	RESULT	FALSE
		TRUE
		CANCEL

RECVFILE

Receive a Petit Computer file another user has saved on their Nintendo DSi system (displays confirmation message).

Format	RECVFILE "resource name: file name"	
Parameters	Resource name	*Refer to LOAD
Error	RESULT	FALSE
		TRUE
		CANCEL

The example of use (P62 Exchanging Files With Other DSi and 3DS Users)

SENDFILE

Send files to another user who has a Nintendo DSi with Petit Computer saved on it (displays confirmation message).

Format	SENDFILE "resource file: file name"	
Parameters	Resource name	*Refer to LOAD
Error	RESULT	FALSE
		TRUE
		CANCEL

The example of use (P62 Exchanging Files With Other DSi and 3DS Users)

05 Drawing Commands

VISIBLE、COLINIT、COLSET、COLREAD()、CHRINIT、CHRSET、CHRREAD()

Display,Draw,Color,Character,etc・・・Commands for visible or seeing.

VISIBLE

Control of screen display elements (using 0 will cause a particular element not to be displayed, while using 1 will display it).

Format	VISIBLE console, panel, BG0, BG1, SPRITE, graphic	
Parameters	Console	0=OFF, 1=ON
	Panel	0=OFF, 1=ON
	BG0	0=OFF, 1=ON
	BG1	0=OFF, 1=ON
	Sprite	0=OFF, 1=ON
	Graphic	0=OFF, 1=ON

The example of use (P49 Resetting Memory and Screen,P85 EXAMPLE 4-04,etc)

COLINIT

Restores the initial color.

Format	COLINIT "color bank name", color number	
Parameters	Color bank name Strings designating target: **BG** BG screens **SP** Sprites **GRP** Graphics	
	Color number	0～255

COLSET

Assign new color

BG number 0 is the background color

Format	COLSET "color bank name", color number, "color data string"	
Parameters	Color bank name	*Refer to COLINIT
	Color number	*Refer to COLINIT
	Color data string	In hexadecimal (base 16) notation(the order is RRGGBB) Each element will be 00~FF(e.g.)""FF00AA""

EXAMPLE:Change BG Screen color number 0.

```
COLSET "BG",0,"8080FF"
```

COLREAD()

Retrieves designated color data. Each element 0~255

Format	COLREAD("color bank name", color number), R, G, B	
Parameters	Color bank name	*Refer to COLINIT
	Color number	*Refer to COLINIT
	R	Variable for red
	G	Variable for green
	B	Variable for blue

EXAMPLE:Display sprite color number 10.

```
0001 COLREAD("SP",10),R,G,B
0002 PRINT R,G,B
```

CHRINIT

Resets designated character to initial state.

Format	CHRINIT "character name"
Parameters	Character name. Strings designating character: BGU0~BGU3 User BG character SPU0~SPU7 User sprite character

CHRSET

Define a single character (8x8 pixel units)

Format	CHRSET “character name”, character number, “graphic string”	
Parameters	Character name	*Refer to CHRINIT
	Character number	0~255
	Graphic string	16 color 8x8 pixel character data is expressed in hexadecimal (base 16) notation (EXAMPLE) "AABBCCDD11223344AABBCCDD11223344AABBCCDD11223344AABBCCDD11223344" (each character expresses 1 pixel)

CHRREAD()

Retrieves data for the designated character.

Format	CHRREAD("character name", character name), C$	
Parameters	Character name	*Refer to CHRINIT
	Character number	*Refer to CHRSET
	C$	Variable for graphic string *Refer to CHRSET

The example of use (P23 EXAMPLE 3-04,etc)

EXAMPLE:Get the deta of BGU0 number 1 character.

```
0001 CHRREAD("BGU0",1),C$
0002 PRINT C$
```

06 Sprite Commands

SPPAGE、SPSET、SPCLR、SPOFS、SPCHR、SPANIM、SPANGLE、SPSCALE、SPCHK()

The following commands allow you to perform actions such as starting and pausing sprite movement:

SPPAGE

Designates the screen to be used for sprites.Although the lower screen can be selected, it is generally used for the keyboard. User characters cannot be displayed on this screen, and only the simple graphics already pre-loaded can be used.(Sprite characters of lower screen are on page 14)

Format	SPPAGE screen	
Parameters	Screen	0=Upper Screen 1=Lower Screen

The example of use (P49 Resetting Memory and Screen,P83 EXAMPLE 4-02,etc)

SPSET

Sprite definition (activation).Activates a sprite designated by a control number. The coordinates are reset to 0,0. Once a sprite has been activated using SPSET and you wish only to change the character number, use the SPRCHR command.

Format	SPSET control number, character number, palette number, horizontal rotation, vertical rotation, order of precedence	
Parameters	Control number	0～99
	Sprite character number	0～511 (for display on lower screen 0～117)
	Palette number	0～15
	horizontal rotation	0=None R 1=Horizontal rotation R (Horizontal rotation=1, Vertical rotation=1 R)
	vertical rotation	0=None R 1=Vertical rotation R
	Order of Priority	The order of priority for sprite display is determined by the control number: the sprite with the lower number will be displayed further forward.

0	In front of console
1	In front of BG (front layer)
2	Between 2 BG layers
3	Behind rear BG layer

The example of use (P81 EXAMPLE 4-01,etc)

EXAMPLE:Assign the sprite whose character number is 0 and palette number is 2 to control number 0

```
SPSET 0,0,2,0,0,0
```

SPCLR

Erase sprite (Prevent sprites being activated)

Format	SPCLR control number	
Parameters	Control number	0~99 (if omitted, all sprites will be erased)

The example of use (P49 Resetting Memory and Screen,P83 EXAMPLE 4-02,etc)

SPOFS

Changes sprite coordinates.

Format	SPOFS control number, x coordinate, y coordinate {,interpolation time]	
Parameters	Control number	0～99
	x coordinate	-1024~+1024(Out of useful range is not error)
	y coordinate	-1024~+1024(Out of useful range is not error)
	Interpolation time	Time taken to automatically add interpolation between current state and new value. (1=1/60th sec)

The example of use (P81 EXAMPLE 4-01,etc)

EXAMPLE:Move the sprite whose control number is 0 to the set coordinate.

```
0001 SPSET 0,0,2,0,0,0
0002 SPOFS 0,255,0,60
```

SPCHR

Changes the sprite character number.

Format	SPCHR control number, character number [, palette number, horizontal rotation, vertical rotation, order of precedence]	
Parameters	Control number	0～99
	Sprite character number	0~511 (for display on lower screen 0~117)
	Palette number	0～15
	horizontal rotation	0=none,1=rotate
	vertical rotation	0=none,1=rotate
	Order of precedence	0～3

SPANIM

Displays sprite animation. Starting with the current designated character number, using this command will make the character number change at defined intervals, It will change within the range of the designated number of frames.

Format	SPANIM control number, number of frames, time [, loop]	
Parameters	Control number	0～99
	Number of frames	1～
	Time	Time to display 1 frame (1=1/60th sec)
	Loop	0=Endless loop, 1～ (Loop number)

The example of use (P81 EXAMPLE 4-01,P83 EXAMPLE 4-02)

EXAMPLE:Animate the sprite set by SPSET.

```
0001 SPSET 0,64,2,0,0,0
0002 SPANIM 0,4,10
```

SPANGLE

This is used to modify the angle of sprites. From the initial position, the start point for rotation will be at the top left of the sprite. Sprite number 0~31 cab be used.

Format	SPANGLE control number, angle [, interpolation time, change direction]	
Parameters	Control number	0~31
	Angle	0~360 (Values outside valid range may be used)
	Interpolation time	Time taken to automatically add interpolation between current state and new value. (1=1/60th sec)
	Change direction	1=Clockwise -1=Anticlockwise (if omitted, will be clockwise)

The example of use (P83 EXAMPLE 4-02)

EXAMPLE:Modify the angle of sprite with control number 0.

```
SPSET 0,64,2,0,0,0
SPANGLE 0,45,60,1
```

SPSCALE

Changes the scaling of sprites. Sprite number 0~31 cab be used.

Format	SPSCALE control number, scale [, interpolation time]	
Parameters	Control number	0~31
	Scale	0~200 (proportion in percent)
	Interpolation time	Time taken to automatically add interpolation between current state and new value. (1=1/60th sec)

The example of use (P83 EXAMPLE 4-02)

SPCHK()

Automatically retrieves interpolation data.

Format	Variable = SPCHK(control number)	
Parameters	Control number	0~99
Returns	State	FALSE=Interpolation is finished. TRUE=Interpolation is in progress.

EXAMPLE:Change the scaling of sprites continuously.

```
SPSET 0,64,2,0,0,0
SPSCALE 0,200,60          ←Change the scale of sprite with control number 0 to 200%.
@1
IF SPCHK(0)==TRUE THEN GOTO @1          ←Wait for interpolation
SPSCALE 0,50,60                         ←Change to 50%
```

07 BG Screen Commands

BGPAGE、BGCLIP、BGOFS、BGPUT、BGREAD()

The following commands allow you to perform tasks such as designating the BG screen to be controlled, altering display offsets and writing onto assigned BG screen positions:

BGPAGE

Designates the BG screen to be controlled.

Format	BGPAGE screen	
Parameters	Screen	0=Upper Screen 1=Lower Screen

The example of use (P84 EXAMPLE 4-03)

BGCLIP

Assigns display parameters (for all layers).

Format	BGCLIP x start point , y start point, x end point, y end point	
Parameters	x start point	0～31
	y start point	0～23
	x end point	0～31
	y end point	0～23

The example of use (P84 EXAMPLE 4-03)

BGOFS

Alters offset of BG screen display.・・・

Format	BGOFS layer, x coordinate, y coordinate [, interpolation time]	
Parameters	Layer	0=Foreground 1=Rear
	x coordinate	x coordinate(0-511) ※Out of useful range is not error
	y coordinate	y coordinate(0-511) ※Out of useful range is not error
	Interpolation time	Time taken to automatically add interpolation between current state and new value. (1=1/60th sec)

The example of use (P84 EXAMPLE 4-03)

BGPUT

Writes onto designated location on BG screens.

Format	BGPUT layer, x coordinate, y coordinate, character number, palette number, horizontal rotation, vertical rotation	
Parameters	Layer	0=front 1=back
	x coordinate	x coordinate(0-63) ※Out of useful range is not error
	y coordinate	y coordinate(0-63) ※Out of useful range is not error
	Character number	0～1023
	Palette number	0～15
	horizontal rotation	0=None,1=rotation
	vertical rotation	0=None,1=rotation

The example of use (P84 EXAMPLE 4-03)

BGREAD()

Obtains data from designated location on BG screens.

Format	BGREAD(layer, x coordinate, y coordinate), CHR, PAL, H, V	
Parameters	Layer	0=Foreground,1=Rear
	x coordinate	x coordinate(0-63) ※Out of useful range is not error
	y coordinate	y coordinate(0-63) ※Out of useful range is not error
	CHR	Variable for character number
	PAL	Variable for palette number
	H	Variable for horizontal rotation
	V	Variable for horizontal rotation

The example of use (P69 Single-Screen Programming③)

08 Graphic Commands

GPAGE、GCOLOR、GCLS、GSPOIT()、GPSET、GPAINT、GLINE、GBOX、GFILL、GCIRCLE、GPUTCHR

The following commands perform tasks such as designating the graphic page,erasing graphics,etc.

GPAGE

Designates the graphic screen to be used.

Format	GPAGE screen	
Parameters	Screen	0=Upper Screen 1=Lower Screen

The example of use (P49 Resetting Memory and Screen,P57 EXAMPLE 3-05)

GCOLOR

Assigns graphic color on graphic screen.

Format	GCOLOR color number	
Parameters	Color number	0～255

GCLS

Erases images on designated graphic screen.

Format	GCLS [color]	
Parameters	Color	0~255(To use color of GCOLOR when it is abbreviated.)

The example of use (P49 Resetting Memory and Screen,P58 EXAMPLE 3-06)

GSPOIT()

Checks color of designated location.

Format	Variable=GSPOIT (x coordinate, y coordinate)	
Parameters	x coordinate	0～255 (Out of useful range is not error)
	y coordinate	0～191 (Out of useful range is not error)
Returns	Color	0~255 (-1 if outside range)

The example of use (P57 EXAMPLE 3-05)

GPSET

Adds a dot.

Format	GPSET x coordinate, y coordinate [,color]	
Parameters	x coordinate	0～255 (Out of useful range is not error)
	y coordinate	0～191 (Out of useful range is not error)
	Color	0~255(To use color of GCOLOR when it is abbreviated.)

The example of use (P57 EXAMPLE 3-05,P58 EXAMPLE 3-06)

GPAINT

Fills in color from designated point. To save time, any color adjacent to the designated location which has the same color will be filled in.

Format	GPAINT x coordinate, y coordinate [,color]	
Parameters	x coordinate	0～255 (Out of useful range is not error)
	y coordinate	0～191 (Out of useful range is not error)
	Color	0~255(To use color of GCOLOR when it is abbreviated.)

GLINE

Draws a line.

Format	GLINE x start point, y start point, x end point, y end point [,color]	
Parameters	x start point	0～255 (Out of useful range is not error)
	y start point	0～191 (Out of useful range is not error)
	x end point	0～255 (Out of useful range is not error)
	y end point	0～191 (Out of useful range is not error)
	Color	0~255(To use color of GCOLOR when it is abbreviated.)

The example of use (P58 EXAMPLE 3-06)

GBOX

Draws a box.

Format	GBOX x start point, y start point, x end point, y end point [,color]	
Parameters	x start point	0～255 (Out of useful range is not error)
	y start point	0～191 (Out of useful range is not error)
	x end point	0～255 (Out of useful range is not error)
	y end point	0～191 (Out of useful range is not error)
	Color	0~255(To use color of GCOLOR when it is abbreviated.)

GFILL

Fills in color of a rectangle.

Format	GFILL x start point, y start point, x end point, y end point [,color]	
Parameters	x start point	0～255 (Out of useful range is not error)
	y start point	0～191 (Out of useful range is not error)
	x end point	0～255 (Out of useful range is not error)
	y end point	0～191 (Out of useful range is not error)
	Color	0~255(To use color of GCOLOR when it is abbreviated.)

The example of use (P57 EXAMPLE 3-05)

GCIRCLE

Draws a circle.

Format	GCIRCLE x coordinate, y coordinate, radius [,color] [, initial angle, final angle]	
Parameters	x coordinate	0～255 (Out of useful range is not error)
	y coordinate	0～191 (Out of useful range is not error)
	Radius	0～255 (Out of useful range is not error)
	Color	0～255(To use color of GCOLOR when it is abbreviated.)
	Initial angle	0～360 (Out of useful range is not error)
	Final angle	0～360 (Out of useful range is not error)

The example of use (P58 EXAMPLE 3-06)

GPUTCHR

This displays the assigned character graphic data on the graphic screen.
This command will copy the data for the designated palette number to the graphic palette. The assigned location will be colored with the 16th shade of the 16th color from the designated palette number.

Format	GPUTCHR x coordinate, y coordinate, "character name", number, palette number, scale	
Parameters	x coordinate	0～255 (Out of useful range is not error)
	y coordinate	0～191 (Out of useful range is not error)
	Character name	*Refer to CHRINIT
	Number	Character number(0~255)
	Palette number	Character color(0~15)
	Scale	Rate of scaling(1,2,4,8)

The example of use (P58 EXAMPLE 3-06)

EXAMPLE : Display BGU0 character on the center of screen.

```
0001 GPUTCHR 128,96," BGU0",16,2,1
```

09 Audio Commands — BEEP、BGMPLAY、BGMSTOP、BGMCHK()

The following commands relate to playing sound effects and background music.

BEEP

Plays a simple warning sound effect.

Format	BEEP [waveform number [,pitch [,volume [,panpot]]]]	
Parameters	Waveform number	0～69 (if omitted, number is 0)
	Pitch	-8192 plays sound 2 octaves lower, 0 is the original sound, 8192 plays it 2 octaves higher. C＝P×0　F#＝P×6　C#＝P×1　G＝P×7　D＝P×2　G#＝P×8　D#＝P×3　A＝P×9　E＝P×4　A#＝P×10　F＝P×5　B＝P×11
	Volume	0=No sound　127=Maximum
	Panpot	0=from left　64=from center　127=from right

The example of use (P61 EXAMPLE 3-09,EXAMPLE 3-10,etc)

● Waveform number

00	Beep
01	Noise
02	Cursor Movement
03	Confirm
04	Cancel
05	Ascend
06	Descend
07	Coin
08	Jump
09	Land
10	Fire
11	Damage
12	Metal
13	Explosion
14	Scream
15	Brake
16	Banjo
17	Synth Strings

18	Synth Brass
19	Synth Bass
20	Guitar
21	Organ
22	Piano
23	Cow Bell
24	Tom-Toms
25	Cymbals
26	Open High-Hat
27	Closed High-Hat
28	Handclap
29	Rimshot
30	Snare Drum
31	Bass Drum
32	OK2
33	BALL
34	Japan Style
35	VOLT

36	AUTO
37	SHOCK
38	ESC
39	Banjo 2
40	Scratching
41	Guitar 2
42	Organ 2
43	Piano 2
44	PASS
45	UP2
46	Record
47	Synth Tom-Toms
48	Cow Bell 2
49	metro
50	tri
51	Conga
52	Dance BD
53	Dance SD

54	Dance HH
55	Hit
56	Timpani
57	Chinese Cymbal
58	Mini Cymbal
59	Shaker
60	Bell
61	Japanese Drum
62	Synthesizer
63	Canorus
64	Puff!
65	Nohkan
66	Humandr1
67	Humandr2
68	Dog
69	Cat

BGMPLAY

Starts playing song. Can play only song in the software.

Format	BGMPLAY [track number,] song number [, track volume]	
Parameters	Song number	0～29

The example of use (P60 EXAMPLE 3-08)

● Song number 30 songs (0～29)

00	Jolly and Jaunty	10	Staff List 2	20	BAL_2
01	Dark and Dank	11	Classical Drama	21	Espionage
02	Tension is Rising	12	Marching Band	22	SCI
03	Upbeat Emotion	13	Ultra-hard Rock	23	Shooting Song
04	Opening Jingle	14	Jolly and Jaunty 2	24	Pad
05	Clear Jingle	15	WOND	25	SEN
06	Game Over	16	Deep in Thought	26	Pure
07	Menu Select	17	WOND2	27	ROA
08	Result Screen	18	For the Future	28	CUR
09	Staff List	19	BAL	29	FIG

BGMSTOP

Stops playing song.

Format	BGMSTOP

The example of use (P49 Clear memory,etc)

BGMCHK()

This lets you check on current music status.

Format	Variable=BGMCHK()
Returns	FALSE=Music stopped TRUE=Music playing

10 Panel & Icon Commands

PNLTYPE、PNLSTR、ICONSET、ICONCLR、ICONCHK()

The following commands let you perform tasks like changing panel type, and checking user system icon status or adjusting their settings.

PNLTYPE

Changes the panel type.

Format	PNLTYPE "panel name"
Parameters	Panel name. Strings that select type to be displayed on Lower Screen:

OFF	No panel is displayed
PNL	When there is no keyboard
KYA	English keyboard
KYM	Symbol keyboard
KYK	Kana keyboard

The example of use (P57 EXAMPLE 3-05,etc)

PNLSTR

String Display on Lower Screen. Line breaks will not be added automatically, even when displaying the last line.

Format	PNLSTR x coordinate, y coordinate, "string", palette number	
Parameters	x coordinate	x coordinate(0-31) ※Out of useful range is not error
	y coordinate	y coordinate(0-23) ※Out of useful range is not error
	String	String you wish to be displayed
	Palette number	0～15

The example of use (P71 Single-Screen Programming Corner,etc)

ICONSET

Settings for user system icon characters (or to initiate display).

Format	ICONSET icon position, icon number	
Parameters	Icon position	User System Icon Number(0~3)
	Icon number	Icon Character Control Number(0~63)

ICONCLR

Cancels display of user system icons.

Format	ICONCLR icon position	
Parameters	Icon position	User System Icon Number(0~3)

ICONCHK()

Check current status of user system icons.

Format	Number=ICONCHK()	
Parameters	None	
Returns	Number	-1=not pressed 0~3 (icon location),

11 Text commands ASC()、CHR$()、VAL()、STR$()、HEX$()、MID$()、LEN()

Character string,character code,etc Character commands.

ASC()

Character code of designated character

Format	Variable=ASC(character)	
Parameters	None	
Returns	Number	Character code of designated character

The example of use (P56 EXAMPLE 3-04,P64 Single-Screen Programming Corner,etc)

CHR$()

Returns the character for the designated ASCII code.

Format	Variable$ = CHR$(character code)	
Parameters	Character code	Number ascribed to each character
Returns	Character	Number ascribed to each character

The example of use (P54 EXAMPLE 3-01,P86 EXAMPLE 4-04,etc)

VAL()

Obtains a number from a string.

Format	Variable = VAL(string)	
Parameters	String	String with number
Returns	Number	Number extracted from string

The example of use (P56 EXAMPLE 3-04,etc)

STR$()

Obtain a string from a number.

Format	Variable$=STR$(number)	
Parameters	Number	Number you want to convert to string
Returns	Character	String generated from number

HEX$()

Gives a hexadecimal string from a number.

Format	Variable$ = HEX$ (numerical value [, decimal places])	
Parameters	Number	Number you want to convert to hexadecimal string
Returns	Character	Hexadecimal string generated from number

MID$()

Extracts a string of a designated length starting from the initial position within the target string.

Format	Variable$ = MID$(string, initial position, number of characters)	
Parameters	String	Original string
	Initial position	Initial position of characters(The top of character position is 0)
	Number of characters	Number of characters to be retrieved
Returns	Character	Extracted string

The example of use (P56 EXAMPLE 3-04,P65 Single-Screen Programming Corner,etc)

LEN()

Obtains the number of characters within a string.

Format	Variable = LEN(string)	
Parameters	String	String you want to determine the length of
Returns	Number	Number of characters (every character is counted as 1)

12 Basic Mathematical Functions

FLOOR()、RND()、ABS()、SGN()、SQR()、EXP()、LOG()、PI()、RAD()、DEG()、SIN()、COS()、TAN()、ATAN()

The following mathematical functions allow you to perform tasks including obtaining integers, absolute values, codes and generating random numbers.

FLOOR()

Obtain the integer or whole number.
You can also use the AND command to obtain an integer in a 1 byte range (e.g.) A=A AND &HFF

Format	Variable = FLOOR(number)	
Parameters	Number	Number
Returns	Number	Requested result

The example of use (P54 EXAMPLE 3-01,P84 EXMPLE 4-03.etc)

RND()

Gives a random number up to the designated value.

Format	Variable = RND (maximum number)	
Parameters	Maximum value	Maximum number generated
Returns	Number	Random number from 0~maximum number (not including the maximum value)

The example of use (P81 EXAMPLE 4-01,P82 A Program with a 50% Probability)

ABS()

Obtains an absolute value.

Format	Variable = ABS (number)	
Parameters	Number	Number from which you want to obtain an absolute value
Returns	Number	Absolute value

SGN()

Obtains a code.

Format	Variable = SGN (variable)	
Parameters	Number	Number to check code
Returns	Number	0 or ±1

SQR( )

Obtains the square root of a number.

Format	Variable = SQR(number)	
Parameters	Number	Original numerical value
Returns	Number	Requested result

EXP( )

Looks for the exponent value.

Format	Variable = EXP(number)	
Parameters	Number	Original numerical value
Returns	Number	Requested result

LOG( )

自然対数を求める。

Format	Variable = LOG(variable)	
Parameters	Number	Original numerical value
Returns	Number	Requested result

PI( )

Obtains value of pi(circumference ratio).

Format	Variable = PI()	
Returns	Number	Value of pi (circumference ratio)

RAD( )

Obtain a radian figure from angle data.

Format	Variable = RAD(angle)	
Parameters	Angle	0～360
Returns	Number	Radian figure from angle

DEG( )

Obtains angle data from radian value.

Format	Variable = DEG(radian)	
Parameters	Radian	0～2π
Returns	Number	Angle from radian value

SIN( )

Returns sine value.

Format	Variable = SIN(radian)	
Parameters	Radian	Radian value of angle
Returns	Number	Requested result

The example of use (P83 EXAMPLE 4-02)

COS( )

Returns cosine value.

Format	Variable = COS(radian)	
Parameters	Radian	Radian value of angle
Returns	Number	Requested result

The example of use (P83 EXAMPLE 4-02)

TAN( )

Returns tangent value.

Format	Variable = TAN(radian)	
Parameters	Radian	Radian value of angle
Returns	Number	Requested result

ATAN( )

Obtains the arc tangent value.
<color rgb5=1f0000>Can also be used as a function for determining direction from 2 parameters (Y, X) and displacement. Desired direction=ATAN(destination y-y, destination x-x)</color>

Format	Variable = ATAN(radian)	
Parameters	Radian	Radian value of angle
Returns	Number	Requested result

The example of use (P44 GAME3-Advice of improvement)

Index

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

ア行

カ行

サ行

ハ行

マ行

ラ行

Afterword

Takuya Matsubara

My name is Takuya Matsuhara and I am responsible for writing the main sections of this guide.As a huge fan of Petit Computer, I am delighted to have had the opportunity to be involved in this publication, and to see it come to fruition.

It should be obvious, even from a cursory glance at the contents page, that this is more than a simple instruction manual or strategy guide. I hope my passion for the golden age of BASIC is evident throughout this guide. In truth, I would have liked even more space to explore the history of BASIC... I am incredibly happy that ASCII Media Works chose to publish this guide, in spite of any concern that its contents would limit it to a very specialist readership.

A company called Jolls were responsible for editing this guide. Editing work entails everything from page layout to correcting the text and coming up with the design. But this guide presented particular challenges, involving a great deal of checking DSi screens, and entering data from the computer monitor into the DSi.

I have worked with Jolls before, collaborating on the programming section of My Con BASIC Magazine, and I was very happy to join forces with them once again. It really does feel like BASIC programming has been reborn.

I was very lucky to have received the generous support of SmileBoom, the company behind Petit Computer, and am grateful for the material they provided, as well as for the opportunity to interview the company president. The comic characters who appeared throughout this guide originally appeared on the official Petit Computer website.

I was also fortunate enough to be permitted to use the official name of the game in the title. What's more, a number of legendary figures from the world of BASIC programming were kind enough to contribute to this guide. Readers who remember that era are bound to be impressed by the line up of names who have been interviewed. I have received support from a large number of people in preparing the images and programs used throughout. I would like to take this opportunity to thank everyone who was involved in the creation of this guide.

Nothing would make me happier than to think that this guide will inspire people to get ever more our of Petit Computer.

A Final Bonus Program

```
0001 A$="ｳﾜﾅﾀﾉﾋﾞﾀｳﾜ"
0002 FOR I=0 TO 9
0003 C=ASC(MID$(A$,I,1))
0004 PRINT CHR$(C+1);
0005 NEXT
```

(Jun 2011)

Further Reading

Official Website
This is SmileBoom's official Petit Computer website, introducing users to a new take on BASIC programming. There's a lot of content to explore, including programs contributed by fans of the software, and lessons for beginners.

http://smileboom.com/special/petitcom/

Official Petit Computer Twitter Account

http://twitter.com/PetitComputer

Petit Computer Wiki Compilation
All the Petit Computer info you could hope for, lovingly compiled by fans. It features bug reports and smart programming tips.

http://wiki.hosiken.jp/petc/

Petit Computer Wiki
Another informative site compiled by fans.

http://www46.atwiki.jp/petitcom/

More About this Guide Available Online

Any corrections or FAQs relating to this guide will be made available at this URL:
http://ascii.asciimw.jp/books/books/detail/978-4-04-870671-1.shtml

Copyright

Disclaimers

※ The texture and color of screenshot images featured in this book vary depending on the device the program is running on, and may differ from what you actually see on screen.
※ Prior to publication, this book was carefully checked for any errors, but its accuracy cannot be guaranteed. If any programs featured in it produce syntax errors, please check that the program code has been entered correctly.

BASIC programming has been reborn.

Petit Computer Official Strategy Technic.

Author Takuya Matsubara
Supervision SmileBoom Co, Ltd.
Editing JOLLS INC.
Publisher Kiyoshi Takano
Location ASCII MEDIA WORKS Inc.
Fujimi 1-8-19 Chiyodaku Tokyoto Japan 102-8584
Tel 0570-003030
Agency KADOKAWA GROUP PUBLISHING CO.,LTD.
Fujimi 2-13-3 Chiyodaku Tokyoto Japan 102-8177
Tel 03-3238-8605

Printers Toppan Printing Co., Ltd.

ISBN978-4-04-870671-1 C3004

First published in Japan in 2011 by ASCII MEDIA WORKS INC.,TOKYO.
English translation rights of Digital publishing arranged with ASCII MEDIA WORKS INC.

English Version edited by Intergrow Inc.

Editing & Editorial Assistance JOLLS INC.（Akihiro Sumiyoshi, Hirofumi Takahashi, Masato Morishima, Hiroshi Yamazaki）
Design & DTP JOLLS INC.（Kenichi Kawamoto, Daisuke,Yasuda）
ASCII Editorial Department - Kazuhiro Kawahara